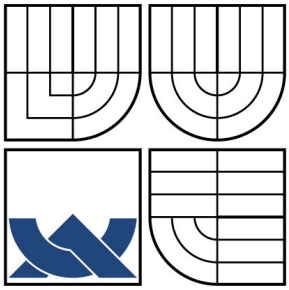
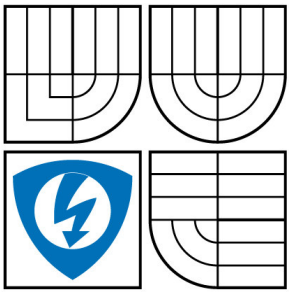


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ VLNKOVÉ TRANSFORMACE PŘI KOMPRESI VIDEOSIGNÁLU

VIDEO COMPRESSION BASED ON WAVELET TRANSFORM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH KINTL

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN MALÝ

SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Kintl Vojtěch, Bc. ***ID:*** 88598
Ročník: 2 ***Akademický rok:*** 2007/08

NÁZEV TÉMATU:

Využití vlnkové transformace při kompresi videosignálu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte současné možnosti komprese videosignálu s přihlédnutím na vlnkovou transformaci (vč. existujících standardů). Dále implementujte jednu z vybraných metod komprese v prostředí MATLAB, včetně testování a simulace.

DOPORUČENÁ LITERATURA:

[1] Misiti M., Misiti Y., Oppenheim G., Poggi J.M.: Wavelet Toolbox for use with MATLAB®, MathWorks, 2002.
[2] Mrak M., Sprljan N., Izquierdo E.: An overview of basic techniques behind scalable video coding, Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium, 2004, pp.597-602.
[3] Kim B.J., Pearlman W. A.: Fast Color-Embedded Video Coding with SPIHT, Proc. 1998 Data Compression Conference, 1998, dostupné z: http://www.cipr.rpi.edu/staff/pearlman.html/papers/dcc98_kp.pdf

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí projektu: Ing. Jan Malý

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Bc. Vojtěch Kintl
Bytem:	Tážaly 90, 783 75, Kožušany-Tážaly
Narozen/a (datum a místo):	20.11.1983, Olomouc

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - ☒ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Využití vlnkové transformace při kompresi videosignálu
Vedoucí/ školitel VŠKP:	Ing. Jan Malý
Ústav:	Ústav telekomunikací
Datum obhajoby VŠKP:	.....

VŠKP odevzdal autor nabyvateli v:

- ☒ tištěné formě – počet exemplářů 2
- ☒ elektronické formě – počet exemplářů 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☒ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……...……
Nabyvatel Autor

ABSTRAKT

Cílem této diplomové práce je nastudovat současné možnosti využití vlnkové transformace při kompresi videosignálu. Část práce je věnována teorii potřebné k praktické realizaci úkolu. Zabývá se popisem videosignálu a jeho vlastností, vlnkovou transformací a metodami komprimace.

Druhá část práce je zaměřena na popis vybrané metody komprese. Jedná se o algoritmus SPIHT (Set Partitioning In Hierarchical Trees), určený pro kompresi statických obrazových dat. Algoritmus je modifikován na použití při kompresi videosignálu, který je charakteristický svou časovou redundancí. Protože algoritmus pracuje v prostorové i časové doméně, je nazýván 3D SPIHT.

Algoritmus je implementován v programovém prostředí MATLAB, který poskytuje důmyslnou podporu vlnkové transformaci (Wavelet Toolbox).

Pro účely snadného a intuitivního ovládání kodéru je vytvořena aplikace, která poskytuje grafické uživatelské prostředí. Uživatel má možnost měnit parametry kodéru a sledovat změny na zobrazovaných náhledech snímků a naměřených grafech. K dispozici jsou čtyři testovací sekvence snímků, které obsahují různé scény, s různými charakteristickými rysy.

Závěr práce je věnován testování navrženého kódovacího schématu, pro různé testovací sekvence snímků a různé nastavené parametry kodéru. Naměřené hodnoty jsou graficky zobrazeny a vyhodnoceny.

KLÍČOVÁ SLOVA

Video komprese, vlnková transformace, SPIHT, 3D SPIHT, MATLAB

ABSTRACT

This diploma thesis focuses on current possibilities concerning the employment of wavelet transformation for video signal compression. One part of the work is devoted to the necessary execution of this task in practise. This deals with video signal and its features description, wavelet transformation and compression methods.

The second part concentrates on description of selected compression method. It is the SPIHT (Set Partitioning In Hierarchical Trees) algorithm which is intended for compression of static image data. The algorithm is modified for usage with video signal compression which is specific for its time redundancy. The algorithm is called 3D SPIHT as it works in the spatial and time domain.

The algorithm is implemented in the MATLAB programming environment which provides a sophisticated support for wavelet transformation (Wavelet Toolbox).
To provide a simple and intuitive encoder control there has been developed an application delivering graphical user interface (GUI). On displayed image previews and measured graphs the user can change encoder parameters and monitor performed changes. There are four image test-sequential modes containing various scenes with different features.

The final part of the work is focused on testing of the proposed encoding scheme, various image test-sequential modes and encoder settings. Measured values are graphically displayed and analyzed.

KEYWORDS

Video compression, wavelet transform, SPIHT, 3D SPIHT, MATLAB

KINTL, V. Využití vlnkové transformace při kompresi videosignálu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 79 s. Vedoucí diplomové práce Ing. Jan Malý.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Využití vlnkové transformace při kompresi videosignálu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . .

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Janu Malému, za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne . . . . . .

. . . . . .
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešní době je samozřejmostí, že téměř každá rodina vlastní televizor, počítač, mobilní telefon, případně i digitální fotoaparát a kameru. Všechny tyto zařízení mají jedno společné, využívají multimediálních dat, jako jsou fotografie, zvuk a video. Dochází k digitalizaci televizního vysílání, implementaci grafických aplikací do mobilních telefonů, pomocí mobilního telefonu je možné fotit nebo nahrávat video. Tato digitální obrazová data jsou velmi populární, zejména proto, že je mnohem jednodušší s nimi pracovat než s analogovými. Digitální přístroje se dají velmi snadno propojit s domácím počítačem, kde je možné jednoduchým způsobem pracovat s pořízenými daty, upravovat fotografie, stříhat video, tisknout, vystavovat fotografie a video na Internetu nebo posílat emailem.

Digitální data je potřeba ukládat. To s sebou nese velké nároky na datové nosiče. Právě toto je u obrazových dat podstatný problém, proto je nutné nějakým způsobem zmenšit jejich velikost, to je dosaženo kompresí. Existuje několik druhů kompresních metod, většinou jsou voleny podle účelu a požadavků na výsledný soubor.

Tato práce je zaměřena na využití vlnkové transformace při kompresi videosignálu. Zejména jsou popsány všechny potřebné metody a vysvětleny pojmy, se kterými se můžeme setkat při kompresi obrazových dat a videa. V neposlední řadě je popsána Diskrétní vlnková transformace s diskrétním časem (DTWT), která je základem kompresní techniky, na kterou se tato práce zaměřuje. Zde jsou také uvedeny některé nejznámější kodeky, které této transformace a jejích vlastností využívají.

Metoda, která je založena na analýze obrazu pomocí banky vlnkových filtrů, je poměrně nová a začala se výrazněji prosazovat např. v podobě grafického formátu *JPEG2000*. Mezi její největší výhody patří především vyšší kvalita obrazu při stejném nebo nižším datovém toku (*bitrate*) ve srovnání s DCT metodou. Další výhodou je, že nevytváří rušivé blokové artefakty v obraze, které se projevují zejména při nízkých datových tocích [7]. Navíc se přímo nabízí po vlnkové analýze využít efektivních cest, jak transformovaná data ukládat, a to například algoritmus *EZW* (Embedded Zerotree Wavelet) [5] nebo *SPIHT* (Set Partitioning In Hierarchical Trees) [6].

Cílem této práce je navrhnout vlastní kódovací algoritmus založený na dvourozměrné diskrétní vlnkové transformaci (2D-DWT) s diskrétním časem a provést jeho testování a simulaci. Při návrhu bylo využito myšlenky algoritmu *SPIHT*, který byl modifikován pro použití při kódování videosignálu. Pro účely snadného a intuitivního ovládání kodéru je vytvořena aplikace, která poskytuje grafické uživatelské prostředí. Uživatel má možnost měnit parametry kodéru a sledovat změny na zobrazovaných náhledech snímků a naměřených grafech. K dispozici jsou čtyři testovací sekvence snímků, které obsahují různé scény, s různými charakteristickými rysy.

Závěr práce je věnován testování navrženého kódovacího schématu pro různé testovací sekvence snímků a různé nastavené parametry kodéru. Získané naměřené hodnoty jsou graficky zobrazeny a vyhodnoceny.

1 CHARAKTERISTICKÉ VLASTNOSTI VIDEA

Jedná se o trojrozměrný signál (3D), z toho jsou dvě dimenze prostorové a jedna časová.

Jeho základní charakteristiky jsou:

- prostorové rozlišení (šířka × výška [pixel])
- obrazová frekvence [obr/s]
- barevná hloubka [bit/pixel]

Tab. 1 : Vlastnosti používaných obrazových formátů [9]

Formát	Šířka [pixel]	Výška [pixel]	Poměr stran obrazu	Obr. frekvence [obr/s]
SQCIF	128	96	4:3	
QCIF	176	144	4:3	5 - 15
CIF	352	288	4:3	10 - 30
4CIF	704	576	4:3	
ITU-R 601 PAL	720	576 (625)	5:4 (4:3)	25
HDV1, HDTV	1280	720	16:9	25, 30, 50, 60
HDV2, HDTV	1440	1080	4:3	25, 30
HDCAM, ATSC	1920	1080	16:9	50

1.1 Prokládané a progresivní snímání

Progresivní snímání obrazu (progressive scanning) snímá obrazové řádky tak, jak jdou po sobě. Nevýhodou je větší šířka pásma (snímkový kmitočet je dvojnásobný oproti prokládanému snímání).

Prokládané snímaní obrazu (interlaced scanning) začalo být používáno v analogových systémech. Výhodou je snížení snímkového kmitočtu a tím šířky pásma, nevýhodou je ale špatná reprodukce horizontálních hran a detailů a rychlých scén [9].

1.2 Intraframe a Interframe komprese

Intraframe znamená kódování uvnitř rámce. Mezirámcová komprese je nazývána ***Interframe***. Kódování uvnitř rámce intraframe využívá redundance informací obsažených v jediném rámci, respektive obrázku. Zkomprimovaný snímek nezávisí na předchozích nebo následujících snímcích. Mezirámcové kódování interframe vychází z nadbytečnosti informací zaznamenaných na více rámcích, porovnáním informací předchozích a následujících obrázků a jejich vyloučením, umožňuje snížit redundanci informace v časové oblasti [9].

1.3 Škálovatelnost videa

V posledním desetiletí vývojové trendy ve vytváření síťových technologií a video kódování dovolují distribuci digitálního obrazového signálu přes různorodou síť, jako je Internet, mobilní bezdrátový systém, televizní vysílání, video na požádání, bezdrátové LAN a další. Potenciální klienti a aplikace mají různé požadavky na obrazovou kvalitu a různé podmínky pro přijímání a dekódování videa.

Škálovatelným videem chápeme video s tak organizovaným datovým tokem, který umožňuje uchovávat video s několika prostorovými, časovými anebo kvalitativními rozlišeními v jednom souboru, dále umožňuje toto video distribuovat přes heterogenní sítě (Internet apod.) koncovým uživatelům dostupnými síťovými prostředky a s různými požadavky na kvalitu zobrazení videa [4].

Tradiční video kódovací systémy, pro účely ukládání, zakódují video sekvence na požadovanou pevnou rychlost přenosu, která je adekvátní pro danou aplikaci. Pokud má sloužit různým klientům, potom aplikace vyžaduje převodník kódu na dané video sekvence. Mimoto, zvláštní aplikace, například Internetový přenos, může dokonce měnit požadavky na rychlost přenosu, během video sekvenčního přenosu. Škálovatelné zakódované video sekvence jsou právě zdrojem pro neomezené množství aplikací bez potřeby převodníku kódů.

Umožňuje různé možnosti vytváření bitového toku:

- Progresivní kódování ve smyslu prostorového rozlišení
- Progresivní kódování ve smyslu přesnosti vyjádření koeficientů

Spočívá v kódování obrazů v různých hladinách:

- Základní hladina – nízká kvalita, nejdůležitější informace
- Rozšiřující hladiny – zvýšení kvality obrazu

Tab. 2 : Módy škálovatelnosti a jejich hladiny [4]

Mód škálovatelnosti	Základní hladina	Rozšiřující hladina
Frekvenční složení	Nízkofrekvenční koeficienty	Vysokofrekvenční koeficienty
Kvalitativní (přesnost vyjádření dat)	Nejvýznamnější bity	Méně významné bity
Prostorové rozlišení	Standardní obrazový formát	Vyšší prostorové rozlišení
Časová dimenze	25 obr/s	50 obr/s

1.4 Kodeky

Video je sekvence po sobě jdoucích obrázků. V počítači se s videem pracuje právě jako s proudem obrázků (snímků), které jsou vhodně digitálně uložené. Nekomprimované video by bylo ale poměrně velké, a proto se komprimuje tzv. kodeky [16].

Kodek (složenina z počátečních slabik slov „kodér a dekodér“, respektive komprese a dekomprese) je zařízení nebo počítačový program, který dokáže transformovat datový proud (stream) nebo signál. Kodeky ukládají data do zakódované formy (většinou za účelem přenosu, uchovávání nebo šifrování), ale častěji se používají naopak pro obnovení přesně nebo přibližně původní formy dat vhodné pro zobrazování, případně jinou manipulaci. Kodeky jsou základní součástí softwaru pro přehrávání multimediálních souborů (hudba, filmy) a často se používají pro videokonference a distribuci multimediálních dat v sítích (streamování).

Síťově šířená multimédia většinou obsahují několik částí, zvuková i obrazová data a navíc doplňující informace (metadata), která umožňují obě složky synchronizovat. Každá z částí může být určena pro jiný program, proces nebo hardware. Aby s nimi bylo možno manipulovat, musí být zapouzdřeny do společného celku.

Další součástí šířených dat může být i obálka, která se, na rozdíl od metadat, nepodílí na informačním obsahu, ale přidává se kvůli zpřístupnění informací nebo pro větší robustnost datového toku. Důvodem je, aby se samotná zakódovaná zvuková a obrazová data odlišila od ostatních součástí datového toku [15].

Kodeky lze rozdělit na ztrátové a bezeztrátové. Bezeztrátové kodeky mají tu výhodu, že video neztratí žádnou informaci. To je ale vykoupeno nízkou kompresí, většinou se kompresní poměr pohybuje okolo 2:1, což je podíl velikosti původních dat ku velikosti komprimovaných dat. Ztrátové kodeky naopak využívají toho, že obraz nemusí být naprosto dokonalý, dokonce může být zkreslený a část obrazové informace se většinou ztrácí. Využívá se různých metod pro odstranění irelevantní složky obrazu, většinou založených na maskovacím jevu lidského zraku.

Různé kodeky se dále liší kvalitou, rychlostí a výslednou velikostí komprimovaného videa [16].

1.4.1 Identifikace formátu videostreamu

Označení videokodeku se skládá vždy ze čtyřmístného kódu, který se nazývá FourCC (Four Character Code), což znamená doslova „čtyř-znakový kód“. Je to sekvence čtyř bajtů užívaná k jedinečné identifikaci formátu dat. Návrh má původ v OSType schématu užívaném v Macintosh systémových programech. FourCC se tedy používá k identifikaci formátu videostreamu a přesně označuje software (kodek) použitý při kódování videodat a tedy i formát těchto dat [12].

1.4.2 Kodeky využívající vlnkové transformace

Pro digitální video bylo vyvinuto a implementováno mnoho algoritmů komprimace. Mezi komerčně nejrozšířenější a nejpoužívanější patří kodeky standardu MPEG (Moving Pictures Expert Group). Skupina MPEG spolupracuje s organizací ISO (International Organization for Standardization), a komisí IEC (International Electro-Technical Commission). Mezi nejznámější standardizované kompresní formáty patří MPEG-1, MPEG-2, a MPEG-4. Hlavním trendem současnosti jsou kodeky založené na standardu MPEG-4, tento standard nedefinuje kódování přímo, ale pouze definuje vlastnosti, které by měl MPEG-4 kompatibilní kodek splňovat. Ze standardu jsou odvozené například kodeky DivX nebo XviD [15].

Další skupinu tvoří kodeky H.261, H.263, pro videokonferenci a H.264 v podobě doporučení organizace ITU-T (International Telecommunications Union - Telecommunications standardization sector). Kodek H.264 vznikl ve spolupráci s organizací MPEG a je také znám jako AVC (Advanced Video Coding) nebo MPEG-4 part10 [15]. Také do této kategorie kodeků patří M-JPEG (Motion-Joint Photographic Experts Group), který kóduje jednotlivé snímky videa kompresí JPEG.

Všechny výše zmíněné kodeky mají společné to, že využívají diskrétní kosinovu transformaci (DCT). Dalšími možnými kompresními algoritmy jsou fraktální metody komprese, které jsou založené na základě fraktální geometrie, ta je dobře využitelná pro kompresi přírodních scenérií s velkou barevnou hloubkou, která ovšem vyžaduje velký výpočetní výkon. Kompresní algoritmy založené na vlnkové transformaci, jsou většinou nasazovány na speciální aplikace nebo jsou součástí výzkumů a testování.

Vlnková analýza se stala moderní disciplínou v oblasti zpracování signálů a v současné době si nachází uplatnění v mnoha výzkumných odvětvích. Díky použití DWT namísto DCT nedochází v obraze k blokovým artefaktům, protože při DWT dochází ke globálnímu zpracování obrazu [1]. Níže jsou uvedeny některé kodeky, které využívají při kompresi diskrétní vlnkovou transformaci (DWT).

Motion-JPEG2000

Standard pro kompresi videa v tzv. intra módu. Je založen na vlnkové transformaci, která je počítána obvykle na celém snímku najednou, což má výhodu v absenci blokového artefaktu.

Jde v podstatě o sekvenci jednotlivých snímků komprimovaných pomocí JPEG2000, více v [2].

Vývojem tohoto standardu se zabývá společnost Morgan, která tento kodek nabízí pod názvem *Morgan M-JPEG2000* (FourCC: MJ2C), aktuální je verze 2, která podporuje HDTV, prokládaný i progresivní režim. Kodek je hardwarově přizpůsobený a optimalizovaný pro Hyper-Threading Technologii a systémy s dvěma CPU [20].

Dirac

Dirac je prototypový algoritmus pro zakódování a dekódování *raw* videa. Byl představen společností BBC v lednu 2004 jako základ nového kodeku pro přenos videa přes Internet. Kodek je pojmenován na počest britského vědce Paula Diraca.

Kodek ještě není finální, stále je vyvíjen. Záměr je, aby byl schopen dekódovat standardní digitální PAL TV (720 × 576i pixel na snímek, 25 snímků za sekundu) v reálném čase. Momentální implementace může dekódovat kolem 17 snímků za sekundu na 3 GHz PC, ale plánují se nová vylepšení.

Dirac je podobný jako běžné video kodeky, jako ISO/IEC Moving Picture Experts Group (MPEG) MPEG-4 a Microsoft WMV 7, které mohou komprimovat jakoukoliv velikost obrazu od nízkého rozlišení QCIF (176x144 pixel) až po HDTV (1920x1080). Nicméně, nabízí významné úspory v šířce pásma a zlepšení obrazové kvality, které předčí i nejnovější generaci kodeků jako je H.264/MPEG-4 AVC nebo SMPTE VC-1 (který je založen na WMV 9 od Microsoftu). Používá vlnkovou kompresi, namísto DCT používající ve starších kodecích. Má dvakrát nižší rychlost přenosu (bit rate) než MPEG-2 HD (High Definition) video.

Dirac je jedním z několika projektů pokoušejících se použít „vlnky“ ke kompresi videa. Vlnková komprimace není pouze experimentální. Jako úspěšná se ukázala ve standardu JPEG2000.

Hlavní výhodou vlnkově založeného kódování je její vlastnost škálovatelnosti (viz. kapitola 1.3).

Indeo

Video kodek původně vyvinutý firmou Intel v roce 1992, ale v roce 2000 ho převzala firma Ligos. Nejposlednější verze poskytuje dobrou obrazovou kvalitu a mnoho různých rychlostí přenosu. Díky vlnkově založené komprimaci nabízí lepší vizuální kvalitu a to na všech úrovních video kvality než předchozí generace kodeků. Originální Indeo kodek byl vysoce asymetrický, to znamená, že k zakódování video streamu potřeboval daleko více času než k jeho dekódování [21].

Scalable Video Coding (SVC)

V současnosti vyvíjený standard pro škálovatelnou kompresi videa, označován také jako MPEG-4 Scalable Video Coding (SVC). Je založen na standardu H.264. Jedná se o jeho rozšíření zejména o časovou pohybově kompenzovanou filtraci (Motion Compensated Temporal Filtering, MCTF), která zprostředkovává časovou škálovatelnost. Zjednodušeně lze tento algoritmus chápat jako vlnkovou transformaci v čase počítanou metodou tzv. liftingu prováděnou na celých snímcích [4].

2 PRINCIPY KOMPRESE VIDEOSIGNÁLU

Cílem komprese je redukovat objem dat za účelem jejich přenosu, archivace nebo ochrany před viry. Přičemž měřítkem kvality může být rychlost komprese, symetrie/asymetrie kompresního algoritmu nebo kompresní poměr. Symetrické algoritmy jsou takové, které potřebují stejný čas pro kompresi i dekompresi. U asymetrických se tento čas liší [11].

Podle principu lze rozdělit komprese na [11]:

- Jednoduché: založené na kódování opakujících se posloupností znaků (RLE).
- Statistické: založené na četnosti výskytu znaků v komprimovaném souboru (Huffmanovo kódování, Aritmetické kódování).
- Slovníkové: založené na kodování všech vyskytujících se posloupností (LZW).
- Transformační: založené na ortogonálních popř. jiných transformacích (fourierovské, vlnkové, fraktálové) a následném odstranění redundance (nadbytečnosti) dat.

Většina digitálních obrázků obsahuje ve větší či menší míře redundantní čili nadbytečné informace. To znamená, že pokud použijeme efektivní ztrátovou metodu komprese, můžeme dosáhnout značné redukce množství informací, potřebných k přenosu obrázků nebo videa. Redundanci můžeme nalézt na úrovni jednotlivých obrazových prvků, řádků nebo rámců, pokud je scéna statická anebo se pohybují jen některé její části. Potíže s implementací videa jsou způsobovány obrovskými nároky na šíři přenosového pásma pro video data. Rychlost přenosu omezuje prakticky použití videa a způsobuje, že pokud chceme přehrávat video v dostatečné kvalitě v reálném čase, stává se komprimace videa nutností [1].

2.1 Redundance a irelevance obrazu

U videosignálů se vyskytují dva základní typy ***redundance***, a to prostorová a časová. Jednotlivé body obrazu jsou charakterizovány základními parametry, a to jasem, barevným tónem a sytostí, které bývají vyjádřeny jasovým a dvěma chrominančními signály. *Prostorová redundance* vzniká v důsledku vzájemné korelace jednotlivých parametrů sousedních bodů obrazu. K její redukci se používá transformačního kódování, které je tím účinnější, čím větší je vzájemná korelace příslušných parametrů sousedních bodů obrazu, a následná kompresní metoda. U obrazu, jehož jasový signál

má charakter šumového signálu, je vzájemná korelace parametrů sousedních bodů velice nízká [10].

U pohyblivých obrazů se kromě prostorové redundance uplatňuje i *redundance časová*. Ta vzniká v důsledku toho, že parametry jednotlivých bodů obrazu jsou v následujícím i předchozím snímku znatelně korelovány (pokud se ovšem právě skokem nezmění scéna – například střih, atd.). K redukci časové redundance se například využívá predikce pomocí snímků I (*Intra Frame*), P (*Predict Frame*) a B (*Bidirectional Frame*), kombinované s tzv. vektory pohybu [10].

Ke snížení objemu dat u obrazových signálů napomáhá také potlačení ***irelevantní*** složky v obrazovém signálu. Využívá se především maskovacího jevu lidského zraku, který spočívá v omezených schopnostech lidského oka rozeznat jemné prostorové detaily, detaily barevných ploch, atd. Míra redukce irelevance je subjektivní veličinou a její stanovení se provádí statistickým vyhodnocením výsledků hodnocení kvality obrazu velkým množství pozorovatelů. Lze ji nastavit vhodným kódováním koeficientů získaných po transformaci [10].

2.1.1 Pojem redundance

Nebo-li nadbytečnost je definována jako větší množství dat, než je množství nutné pro přenos dané informace, které mohou být z přenosu odstraněny bez toho, aby nebyla poškozena nebo omezena přenášená data vzhledem ke ztrátám v komunikačním kanálu. Je to tedy množství znaků, symbolů nebo bitů v uvažovaném digitálním signálu, které je možné eliminovat, aniž by došlo ke ztrátě informace. Redukce redundance je vratný proces [10].

2.1.2 Pojem irelevance

Je definována jako nepodstatná (zbytečná) složka informace, kterou je možné ve zdrojovém kodéru zcela potlačit a dále již nepřenášet, neboť příjemcem na přijímací straně stejně nemůže být vnímána. Redukce irelevance je nevratný proces, představující ztrátu informace [10].

2.2 Metody komprimace

Metoda komprimace může být ***ztrátová*** a ***bezeztrátová***. Při bezeztrátové komprimaci je možné rekonstruovat dokonale původní signál zdroje informací. Používá se výhradně pro kompresi textů a dat. Naopak ztrátová komprese již neumožňuje dokonalou zpětnou rekonstrukci původního signálu, avšak tyto metody umožňují dosáhnout vyšší komprimační poměry, což je s ohledem na paměťově náročná média, jako je video i zvuk, žádoucí a v mnoha případech nutné. Při ztrátové komprimaci

vycházíme obvykle z modelu vnímání informací uživatelem a podle požadavku na kvalitu, respektive rozsahu vnímání, redukujeme objem přenášené informace. Lidské oko není dokonalé a rozlišovací schopnost zraku má také jistá omezení, která se při komprimaci velmi často využívají [1].

Mezi nedokonalosti lidského zrakového vnímání patří konečná rozlišovací schopnost jasové složky (méně než 200 odstínů šedi), nižší rozlišovací schopnost pro barevné složky, konečná rychlost pozorování změn obrazové scény (méně než 20 snímků za sekundu) a nižší rozlišovací schopnost detailů pro pohybující se objekty [9].

2.2.1 Ztrátová komprese video dat

Jedná se především o potlačení některých dat. V této části kompresního algoritmu je rozhodující kvalitní psychovizuální model, který určuje, jaká data mohou být potlačena nebo dokonce úplně odstraněna. Při kompresi obrazu se posuzuje, které frekvence v obraze jsou důležité, aby člověk na obrázku viděl to, co na něm vidět má [1].

Ztrátovou kompresi, týkající se videosignálu, lze rozdělit na:

- komprese statických obrázků a jejich sestavení do sekvence (M-JPEG, M-JPEG2000),
- komprese na základě predikce mezi snímky (MPEG-1, MPEG-2, H.264),
- komprese pomocí popisu objektů scény (MPEG-4),
- komprese pomocí modelů objektů.

2.2.2 Bezeztrátové komprese

Při bezeztrátové komprimaci se snažíme o snížení redundance obsažené v datech, nebo-li o entropické kódování. Nesmí dojít ke ztrátě informace. Níže jsou uvedeny metody, se kterými je možné setkat se nejčastěji.

Huffmanovo kódování

Toto kódování spočívá v přiřazování kódových slov jednotlivým znakům zprávy na základě statistických vlastností (četnost výskytu) znaku. Četnějším znakům odpovídají kratší kódová slova. Méně četným delší kódová slova. Kódová slova musí vyhovovat tzv. prefixové podmínce. Žádné kratší kódové slovo nesmí být předponou (prefixem) jiného kódového slova. Přiřazování je dáno tabulkou (Huffmanova tabulka). Jelikož se mohou statistické vlastnosti znaků zprávy měnit je výhodné tabulku upravovat, což vede k dynamickému Huffmanovu kódování s vyšší účinností kódování.

Huffmanovo kódování používá kódová slova s proměnnou délkou (Variable Length Coding -VLC) [13].

Algoritmus kódování spočívá v zjištění četnosti jednotlivých znaků v kódovaném souboru, vytvoření binárního stromu (Huffmanova kódu jednotlivých znaků), uložení tohoto stromu a nahrazení symbolů jednotlivými kódy (posloupností bitů).

Kódování délky běhu (Run-Length Encoding - RLE)

Kóduje vstupní data tak, že kóduje posloupnosti stejných symbolů do dvojic (počet_opakování, symbol). Účinnost komprese je silně závislá na charakteru vstupních dat, která musí obsahovat delší sekvence stejných znaků, jinak výrazně účinnost komprese klesá. Často se používá v počítačové grafice pro kompresi obrazu, který obsahuje větší plochy stejné barvy nebo jako pomocná metoda komprese dat, například u standardu JPEG [11].

Aritmetické kódování

Dosahuje vyšší účinnosti než Huffmanovo kódování. Spočívá v kódování celé skupiny znaků zprávy najednou. Skupinu znaků nahradíme podintervalem z intervalu <0, 1) na základě četnosti výskytu jednotlivých znaků. Jakékoliv číslo z podintervalu potom reprezentuje zakódovanou skupinu znaků [13].

Slovníkové kompresní metody

Slovníkové metody komprese jsou často používané pro kompresi dat. Mezi nejznámější patří LZW (Lempel-Ziv-Welch) metoda**.** Princip spočívá ve vyhledávání opakujících se posloupností znaků, ukládání těchto posloupností do slovníku pro další použití a přiřazení jednoznakového kódu těmto posloupnostem. Je to jednoprůchodová metoda, nevyžaduje předběžnou analýzu souboru dat. Při průchodu komprimovaným souborem se vytváří slovník, kde první položky jsou znaky původní abecedy a zbývající položky tvoří posloupnosti znaků souboru obsažené v komprimovaném souboru [11].

2.2.3 Kompresní poměr

Kompresní poměr je podíl velikosti původních dat ku velikosti komprimovaných dat. Například při kompresi 10MB souboru do 2MB je poměr 10/2 = 5 (tj. 5 : 1, což znamená pětkrát zmenšeno). Kompresní poměr je ovlivněn volbou kompresního algoritmu i typem komprimovaných dat [15].

2.3 Transformační kódování

Mějme dvourozměrný (2D) obraz. Na tento obraz lze uplatnit vhodně zvolenou transformaci, která převede obraz do jiné oblasti. V této oblasti je informace obsažená v obraze vyjádřena ve tvaru, který je výhodnější pro následné zpracování (např. odstranění redundance pomocí entropického kódování). Jelikož je obraz 2D, musí být i použitá transformace dvourozměrná. Lze použít např. následující transformace: Diskrétní Fourierovou transformaci (DFT), Diskrétní kosinovou transformaci (DCT), nebo Diskrétní vlnkovou transformaci (DWT) [13].

Po transformaci obrazu, následném libovolném zpracování a přenosu opět převedeme obraz do původní prostorové oblasti inverzní transformací. To znamená, že je zde požadavek, aby k dané transformaci existovala i transformace inverzní.

Výhodou je, pokud je transformace separabilní. To znamená, že ji lze uplatnit nejprve na řádky a poté na sloupce obrazové matice, což umožňuje využití jednorozměrné transformace. Důležité je zdůraznit, že by transformace měla být sama o sobě bezeztrátová. Pouze připravuje signál k dalšímu zpracování [13].

3 VLNKOVÁ TRANSFORMACE

Vlnková transformace (Wavelet transform) je způsob, jak v libovolném signálu (např. zvuku, obrazu, signálu ze senzoru) rozlišit jednotlivé komponenty, ze kterých je signál složen a ty vhodně zobrazit. Vlnková transformace je výpočetní algoritmus vhodný pro implementaci na DSP (signálovém procesoru) nebo v PC, založený na porovnávání analyzovaného signálu s předem vybraným krátkým vzorem průběhu, tzv. vlnkou. A právě podobnost vlnky s původním signálem pro různá posunutí a roztažení vlnky je hledaným výsledkem transformace [3].

Vlnková transformace se dá využít k celé řadě aplikací, například odstranění rušivých složek nebo právě ona zmíněná komprese. Po rozložení signálu do komponent lze provést odstranění/vymazání složek s malým vlivem a opětovným složením tak získat digitální signál s menším množstvím dat. Proto se často používá k odšumování a odrušování libovolných signálů, kompresi signálů a to i v reálném čase (zvuku, hudby, obrazu, videa), rozpoznávání hran a obrysů v obraze nebo k zvýrazňování určitých částí signálů [3].

3.1 Princip vlnkové transformace

Celý princip základního výpočetního algoritmu spojité vlnkové transformace CWT (Continuous Wavelet Transformation) spočívá ve vzájemném porovnávání signálu *x(t)* se zvoleným vzorovým tvarem, označovaným jako vlnka ψ (wavelet). Výsledkem porovnání je koeficient - hodnota *CWT (τ, s)*, udávající úroveň podobnosti vzoru-vlnky se signálem.

CWT lze matematicky vyjádřit následujícím vztahem [3]

$$CWT(\tau,s)=\frac{1}{\sqrt{|s|}}\int x(t)\psi\left(\frac{t-\tau}{s}\right)dt\,, \tag{3.1}$$

kde τ je časové posunutí a s změna měřítka.

Umožňuje přesně lokalizovat jak průběh komponent signálu v časovém měřítku, tak i ve frekvenčním. A právě to je přínos Vlnkové (Wavelet) transformace.

3.2 DWT (Diskrétní Vlnková Transformace)

DWT lze chápat jako speciálně vzorkovanou spojitou vlnkovou transformaci CWT, kterou lze počítat rychlým algoritmem, tvořeným filtrací FIR filtry a podvzorkováním (decimací) [7].

Protože použití spojité vlnkové transformace by nám přineslo teoreticky nekonečné množství dat, z nichž naprostá většina je pro nás zcela nezajímavá,

využíváme v praxi diskrétní vlnkovou transformaci (DWT). Ve skutečnosti se jedná o DTWT diskrétní vlnkovou transformaci s diskrétním časem, DWT je totiž definována obecněji, pro spojitá data. Vzhledem ke zjednodušení budeme ale uvažovat v dalším textu DWT jako čistě operaci s diskrétními daty a diskrétním časem. Ta si z celého spektra transformovaných informací vybírá pouze měřítka a translace tzv. *dyadické* (to znamená, že každé dílčí pásmo je polovinou (u 2D čtvrtinou) předchozího pásma).

Lze ji vyjádřit pomocí ortogonální matice $\boldsymbol{W}$ řádu $n \times n$. Je-li $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ vektor délky n, pak jeho waveletovou transformací je vektor $\boldsymbol{d}=(d_1, \ldots, d_n)^T$, získaný jako

$$d = Wy . \tag{3.2}$$

Inverzní waveletová transformace je díky ortogonalitě $\boldsymbol{W}$ vyjádřena jako

$$y = W^{-1} \cdot d = W^T \cdot d \quad . \tag{3.3}$$

Z předchozího vyplývá důležitá vlastnost vlnkové transformace – linearita.

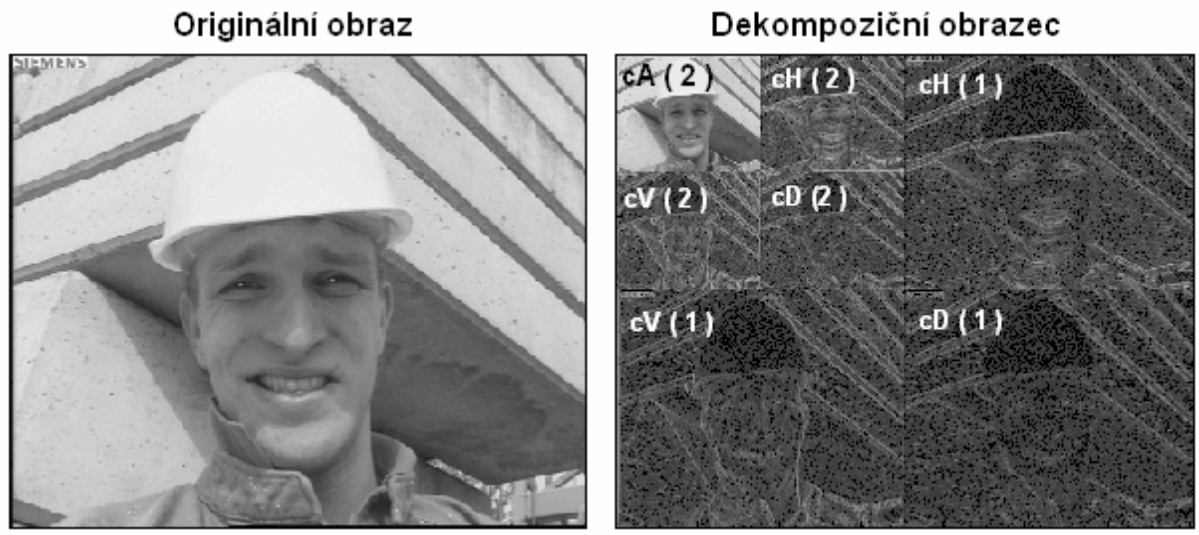
Nejpoužívanější praktická realizace DWT spočívá ve struktuře páru kvadraturních zrcadlových filtrů (QMF) tvořených dolní propustí DP (scaling filter) a horní propustí HP (wavelet filter), které mají komplementární propustná pásma. Základem dolní propusti je měřítková funkce a z ní vypočítaný měřítkový filtr *w*, z kterého normováním vzniknou koeficienty filtru dolní propusti *h*. Koeficienty horní propusti *g* pak vzniknou inverzí impulsní funkce dolní propusti *h*. Výstupy obou filtrů jsou pak decimovány 2, tj. podvzorkovány na polovinu vstupních vzorků. Výsledkem filtrace každého stupně je posloupnost koeficientu/vzorků tzv. detailů DWT z větve s horní propustí HP a posloupnost koeficientů/vzorků tzv. aproximace z větve s dolní propustí DP. Díky decimaci je celkový počet koeficientů po jednom kroku stejný jako počet vstupních vzorků. Koeficienty aproximace lze dále analyzovat dalším stupněm banky filtrů a obdržet tak další soubor koeficientů aproximace a detailů. Tak lze postupovat až do úplného vyčerpání informace ze vstupního signálu, to znamená, že již nelze žádná další detaily nalézt. Zajímavé je, že lze posloupnost detailů z každého stupně zpět převést do časové oblasti na signál stejné struktury jako na vstupu, pouze s tím rozdílem, že zde bude obsahovat jen příslušnou komponentu odpovídající daným detailům [3].

Jednotlivé složky, které nechceme, aby signál obsahoval, nebo mají takový malý vliv (amplitudu), že jsou jen stěží rozpoznatelné, lze například úplně zahodit a jejich posloupnost nahradit nulami (nulovým signálem). Pak zpětnou transformací můžeme za pomoci syntetizující části banky filtrů opět spojit do jednoho signálu, která již však nebude obsahovat vynulované složky. V syntetizující části banky filtrů se využívá rekonstrukčních FIR filtrů (Finite Impulse Response – filtry s konečnou impulsní odezvou) s koeficienty *h*' a *g*', které jsou určeny jako časově obrácené sekvence koeficientů dekompozičních filtrů *h* a *g* [3].

Vlnková transformace se v posledních letech více a více rozšiřuje jako kvalitní metoda zpracování některých číslicových signálů. Není však vhodná pro všechny signály. Některé svým tvarem například nevyhovují ani jedné z mnoha nabízených vlnek.

Hodí se zejména pro zpracování hudby a obrazu. Obrazový signál je dvojrozměrný, proto se používá 2D-DWT. Vlnková transformace je separabilní, proto lze zpracovat nejprve sloupce a poté řádky [7]. Tímto způsobem se získají čtyři skupiny koeficientů (aproximační *cA*, horizontální *cH*, vertikální *cV* a diagonální *cD*). Aproximační koeficienty udržují celkovou informaci o obrazu, zatímco detailní koeficienty (*cH, cV a cD*) reprezentují detaily v příslušném směru. Pro názorné zobrazení se používá dekompoziční obrazec. Na *obr. 1* je vidět uspořádání jednotlivých koeficientů.

Pro rekonstrukci je nutné provést nadvzorkování, to se provede doplněním nulami. Poté se rekonstrukčními filtry obraz vrátí do původního stavu.



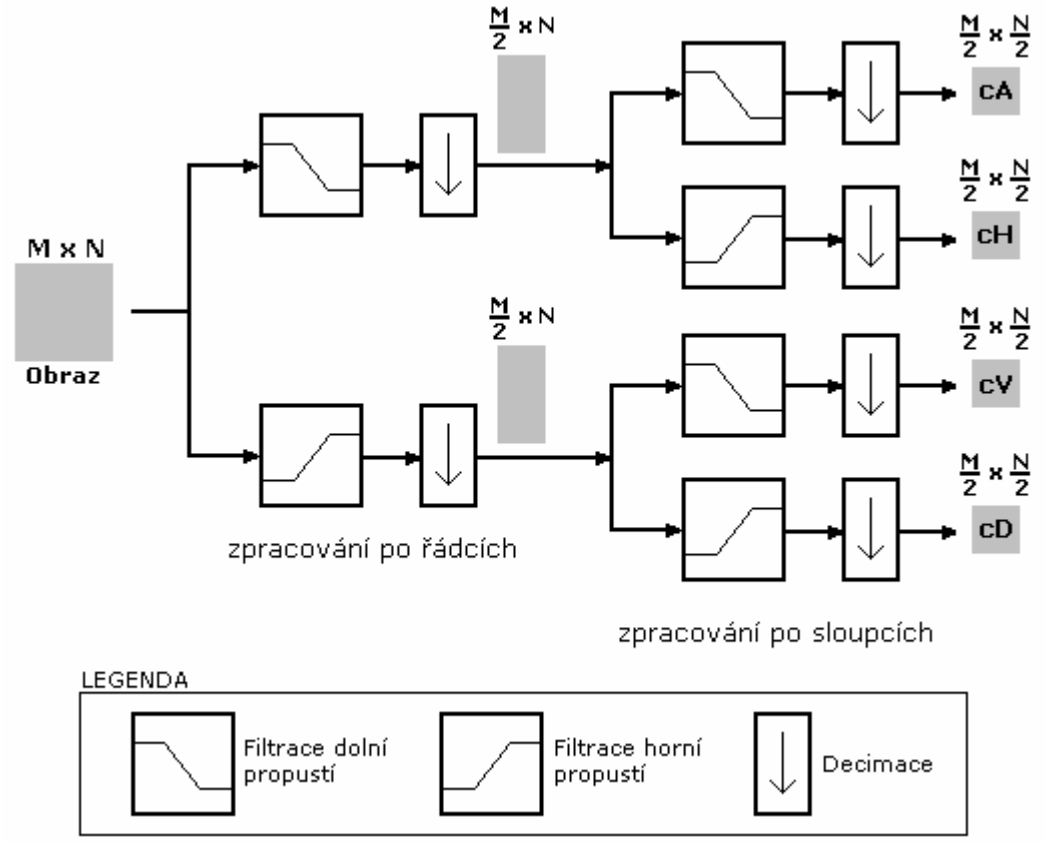
Obr. 1 : Ukázka dekompozičního obrazce hloubky 2, wavelet bior3.1

3.3 Dvourozměrná vlnková transformace

Pro zpracování obrazových dat je potřeba dvourozměrné diskrétní vlnkové transformace, která bývá označována jako 2D-DWT. Protože je vlnková transformace separabilní, lze zpracovat nejprve řádky a poté sloupce. K rozkladu jsou použity řádkové a sloupcové filtry. Celý postup lze popsat několika kroky. Nejprve se filtruje obraz po řádcích a výstup se decimuje a dále se filtruje po sloupcích a výsledek se opět decimuje. Tímto postupem dostaneme čtyři výsledné obrazy, které jsou však poloviční vzhledem k vstupnímu obrazu. Pokud by jsme chtěli provést další krok rozkladu, aplikovali by jsme ho na aproximační koeficienty *cA*. Celý princip vystihuje *obr. 2*.

Rekonstrukce se provede analogicky přesně opačným způsobem, přičemž operace decimace je nahrazena vložením nuly mezi každý prvek (nadvzorkováním) [7].

Pro urychlení celého výpočtu rozkladu se používají různé algoritmy, např. ***Fast Lifting Wavelet Transform,*** který je použit ve standardu JPEG2000. Princip algoritmu spočívá v rozdělení původních filtrů na více menších úseků, tím se dosáhne zmenšení výpočetní náročnosti a snížení paměťových nároků programu [7].



Obr. 2 : Princip rozkladu obrazu pomocí 2D-DWT

3.4 Některé používané vlnky

Existuje kolem 400 používaných vlnek, které jsou více či méně vhodné pro různé úlohy. Vlnky můžeme v podstatě klasifikovat do dvou druhů; orthogonální a biortogonální. Typickými zástupcem orthogonálních vlnek jsou např. vlnky Daubechies. Všechny filtry orthogonálních vlnek jsou stejně dlouhé a mají vždy sudý počet prvků.

Biorthogonální vlnky jsou mnohem více univerzální a poskytují při návrhu filtrů daleko větší volnost, než vlnky orthogonální. Z hlediska filtrace získáme čtyři filtry,

přitom jejich délka nemusí být stejná a nemusejí mít ani sudý počet prvků. Podmínka perfektní rekonstrukce je však vždy dodržena [7].

Ve velké části dostupné literatury popisující aplikace vlnkové transformace se uvádí, že výběr použité vlnky byl proveden zkusmo nebo intuitivně. Bylo však nalezeno několik souvislostí mezi řešenou úlohou (charakterem analyzovaného signálu) a vhodnou vlnkou [3].

3.5 Přednosti vlkové analýzy

Metoda, která je založená na analýze obrazu pomocí banky vlnkových filtrů, je poměrně nová a začala se výrazněji prosazovat např. v podobě grafického formátu *JPEG2000*. Mezi největší výhody kompresí založených na vlnkové analýze obrazu patří především vyšší kvalita obrazu při stejném nebo nižším datovém toku (bitrate) ve srovnání s DCT metodou a dále absence některých velmi rušivých artefaktů v obraze, projevujících se zejména při nízkých datových tocích. Navíc se přímo nabízí po vlnkové analýze využít efektivních cest, jak transformovaná data ukládat, a to například algoritmus *EZW* (Embedded Zerotree Wavelet) [5] nebo *SPIHT* (Set Partitioning In Hierarchical Trees) [6], který hledá optimální poměr mezi kvalitou a výsledným datovým tokem mnohem lépe než při využití klasických kvantovacích metod [7].

3.6 Kódování koeficientů

V *kapitole 3.3* byla popsána dvourozměrná vlnková transformace, jejímž výstupem je dekompoziční obrazec (viz. *obr. 1*). Tato kapitola se zabývá kódováním tohoto dekompozičního obrazce, tedy kódováním koeficientů vlnkové transformace.

3.6.1 Skalární kvantování

Na rozložený dekompoziční obraz, jako je např. na *obr. 1*, může být aplikováno skalární kvantování koeficientů. Jde o základní metodu, která pracuje s jednotlivými koeficienty jako samostatnými celky s ohledem na jedno pásmo dekompozičního obrazce. Kvantizační krok se stanovuje buď globálně nebo výhodněji samostatně pro každou úroveň dekompozice zvlášť. Skalární kvantizace nahradí původní koeficienty (obecně reálné) celočíselnými hodnotami se znaménkem, které vzniknou podělením původního koeficientu tzv. kvantizačním krokem (který se stanoví např. pomocí dynamického rozsahu hodnot v dané úrovni dekompozice). V tomto kroku tedy dochází ke ztrátě informace.

Po kvantizaci je nutné aplikovat na výsledek bezeztrátovou kompresi informace, nejčastěji se pracuje s aritmetickým kódováním, případně je možné využít Huffmanova kódování, pokud je možných hodnot mezi koeficienty více [18].

3.6.2 Vektorové kódování

Při vektorovém kódování se snažíme nalézt vazby mezi jednotlivými prvky i mezi různými úrovněmi dekompozičního obrazce a využít je ve prospěch kódování. Pokud zaměříme pozornost na rozložení vlnkových koeficientů v dekompozičním obrazci, můžeme zaznamenat jistou spojitost mezi stejnými skupinami koeficientů v různých kvantovacích úrovních. Každá skupina detailních koeficientů *cH*, *cV* a *cD* se vzhledem k nejvyšší úrovni dekompozice rozměrově zmenšuje a koeficienty, které nebyly decimací vypuštěny, naopak nabývají větších hodnot. Tento jev je poměrně dobře predikovatelný, neboť každá úroveň je vzhledem k vyšší rozměrově poloviční (důsledek dyadické dekompozice). Pokud se na tento fakt podíváme čistě z programátorského hlediska, můžeme pozorovat určitou stromovou strukturu, každému koeficientu v určité úrovni odpovídá skupina čtyř koeficientů v úrovni o stupeň vyšší [7, 18].

Právě této vlastnosti využívají metody kódování EZW (Embedded Zerotree Wavelet), tvůrcem je J. M. Shapiro, publikováno v roce 1993 [5]. Daleko efektivnější řešení je bitově orientované SPIHT (Set Partitioning in Hierarchical Trees) kódování, se kterým přišli v roce 1996 A. Said a W. A. Pearlman [6], které z velké části vychází právě z EZW a dále jej rozšiřuje. Metody se skládají z kroků, při kterých provádíme srovnávání hodnot koeficientů s určitým prahem, který se s každým dalším krokem snižuje. Vzhledem k využití stromových struktur a předpokladu zvětšování hodnot koeficientů se zvyšující se úrovní dekompozice je velmi pravděpodobné, že velké množství koeficientů ležících pod úrovní prahu se nám podaří zakódovat několika málo bity. Co do efektivity komprese se jedná o jedno z nejlepších řešení, mimo jiné i co se týče otázky dostatečné bitové variace (další komprese vygenerovaného datového toku bezeztrátovou metodou není dokonce vůbec nutná) [7].

Shrneme-li výhody vektorové kvantizace, je to její „vkládaný“ charakter, postupné prahování obrazu v krocích, díky němuž můžeme obraz rekonstruovat postupně již z velmi malého množství informace a v každém dalším kroku přidávat do obrazu další detaily. Druhou výhodou je extrémně nízká velikost komprimovaných dat (to je dáno implementací stromových struktur, které využívají podobnosti jednotlivých koeficientů v po sobě následujících úrovních, a kódováním hodnot relevantních koeficientů v daném kroku pouze několika málo bity) [18].

Jak ukazuje řada výzkumů [2], má JPEG2000 (založený na DWT, využívající vektorového kódování) i při velmi vysokých stupních komprese subjektivně daleko vyšší kvalitu dekódovaného obrazu, než klasický JPEG (založený na DCT), a algoritmická náročnost komprese je přitom pouze nepatrně vyšší [2].

4 KOMPRESNÍ ALGORITMUS SPIHT

Tato kapitola je zaměřena na popis algoritmu SPIHT (Set Partitioning In Hierarchical Trees), ze kterého bylo vycházeno při návrhu vlastní metody komprimace videosignálu založené na vlnkové transformaci.

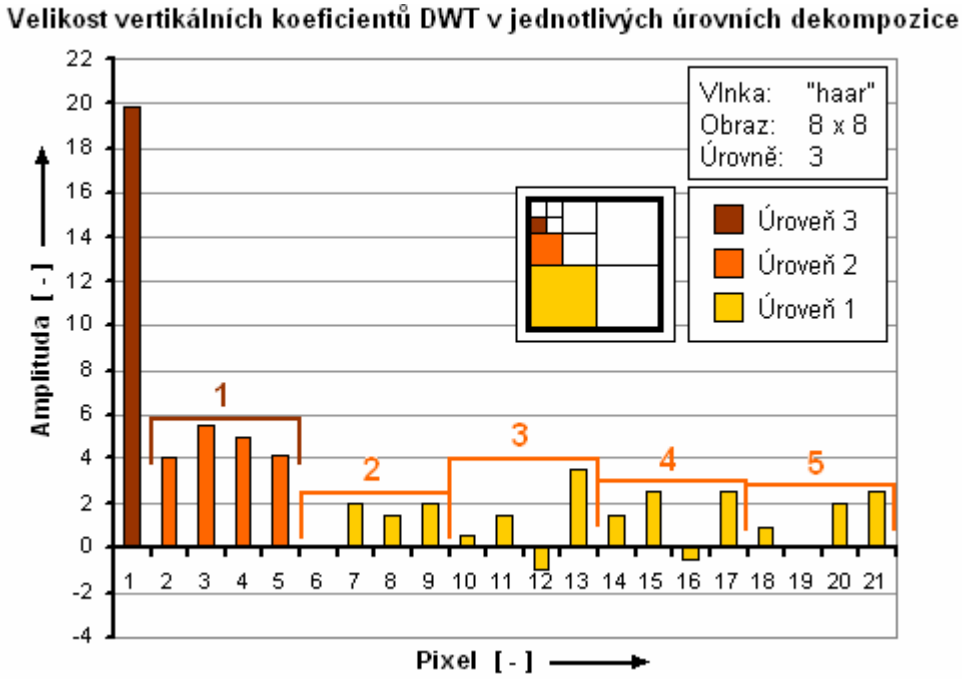
Navržený algoritmus je pojmenován jako 3D SPIHT, protože rozšiřuje původní SPIHT, určený pro kompresi obrazových dat, o další rozměr, a tím je čas (časová závislost jednotlivých snímků ve videu).

4.1 Vlastnost vlnkových koeficientů

Podíváme-li se podrobněji na skladbu vlnkových koeficientů v různých úrovních dekompozice, zjistíme, že se tu nachází jistá spojitost mezi stejnými skupinami koeficientů v různých úrovních dekompozice. Dekompoziční obrazec se skládá z určité stromové struktury stejných obrazových dat, které se s rostoucí úrovní dekompozice zmenšují (dvakrát s každou přibývající úrovní) a informace v nich obsažená se zhustí (počet příslušných vlnkových koeficientů se zmenší a koeficienty získají větší hodnoty).

Tato skutečnost je výstižně ukázána na *obr. 3*, kde jsou ukázány vertikální koeficienty po DWT obrazového výřezu 8×8 pixelů z reálného snímku.

Právě této vlastnosti vlnkových koeficientů se dá s výhodou využít, kdy kompresní algoritmus předpokládá výskyt určitých skupin koeficientů na určitých pozicích a zároveň předpokládá, že jejich velikost se postupně zmenšuje směrem k nižší úrovni dekompozice.



Obr. 3 : Vlastnost vlnkových koeficientů v různých úrovních dekompozice

4.2 Obecný princip algoritmu SPIHT

Algoritmus SPIHT se skládá z několika kroků, které budou postupně vysvětleny.

1. Získání dekompozičního obrazce. Toto bylo popsáno v předchozích kapitolách, pouze je nutné dodat, že předpokládané hodnoty v matici koeficientů jsou v 16-ti bitovém rozsahu včetně znaménka (hodnoty koeficientů se mohou tedy pohybovat v rozmezí $-(2^{15}-1)$ až $(2^{15}-1)$).
2. Kvantizace koeficientů na 0 nebo 1, dle prahu nastaveného jako 2^n (kde *n* je krok kvantizace a je inicializován tak, aby *n* představovalo počet bitů absolutní hodnoty maximálního prvku mínus jedna). Pokud bude koeficient v absolutní hodnotě větší než hodnota prahu 2^n, bude výstupem 1 následovaná znaménkovým bitem, jinak je výstup 0. Čili nenulové hodnoty jsou získány pro koeficienty, jejichž MSB (Most Significant Bit neboli nejvíce významný bit) je 1. Tím vzniká tzv. *signifikantní mapa*, jejíž hodnoty poukazují na *signifikanci* koeficientu, tj. zda je daný koeficient vzhledem k prahovací úrovni důležitý.
3. Snížení prahovací úrovně na 2^{n-1} a celý postup se opakuje, ale pouze na koeficientech, které byly v předchozích krocích označeny jako *insignifikantní*, tedy nedůležité. Vygenerovaný binární kód řadíme za kód předchozího kódu, vzniká tak postupně výsledný bitový tok.
4. Krok 3 se opakuje, přičemž se postupně snižuje úroveň prahu na 2^{n-2} , 2^{n-3} , atd.

Při každém snížení prahu se celkový počet bitů využitý k reprezentaci obrazu zvětšuje, stejně jako kvalita výsledného (dekódovaného) obrazu. Toto je velmi zajímavá vlastnost. Proces kódování / dekódování je možno tedy kdykoliv přerušit a lze získat alespoň určitou informaci o výsledném obrazu. Tato vlastnost se nazývá *progresivní (postupné) kódování*. Mimo jiné je díky ní přímo možné definovat, jaká bude přesná výsledná velikost souboru uložených dat [7].

4.3 Organizace dat v prostorových stromech

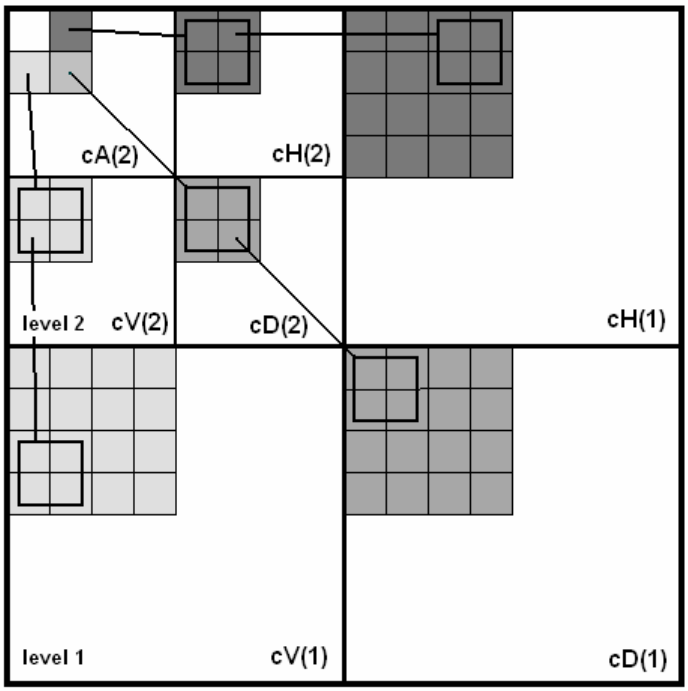
Prostorový strom je základní jednotka organizace dat pomocí SPIHT. Jeho struktura je vidět na *obr. 4*, kde jsou zachyceny na příkladu dvouúrovňového dekompozičního obrazce. Koeficienty v této stromové struktuře jsou organizovány do skupin po 2×2 prvcích (jedna taková skupina je na obrázku). První prvek této skupiny (levý horní bílý) není nikdy uvažován jako kořen stromu. Ostatní tři prvky už mají svoje potomky (tak jsou označovány následovníci v nižších hloubkách stromu). Je to proto, aby každý strom začínal pouze jedním koeficientem (nazývaným kořen stromu). Každý z těchto prvků má čtyři potomky, tedy opět skupina 2×2 prvků.

Tito potomci jsou lokalizováni podle *obr. 4*, a to tak, že v každé nižší úrovni dekompozice je další úroveň (hloubka) prostorového stromu. Tedy, koeficienty v *cA*(2)

budou mít potomky v *cH*(2), *cV*(2) a *cD*(2). Potom koeficienty v *cH*(2) budou mít potomky v *cH*(1) a analogicky i *cV*(2) v *cV*(1) a *cD*(2) v *cD*(1).

Přesná lokalizace potomků vzhledem k nadřazeným prvkům je následující: Každý prvek, který je součástí stromu a má v dekompozičním obrazci souřadnice *(i,j),* bude mít následovníky na souřadnicích *(2i,2j), (2i+1,2j), (2i, 2j+1), (2i+1,2j+1).* Toto pravidlo je v celém algoritmu univerzální a platí pro všechny koeficienty kromě těch, které nejsou kořeny prostorového stromu.

Protože z hlediska principu vlnkové dekompozice není tak těsná souvislost mezi skupinou 2×2 prvků (resp. tří platných kořenů prostorových stromů z této skupiny) z *cA* a jejich potomky v *cH, cV* a *cD* poslední úrovně dekompozice jako mezi stejnými detailními koeficienty různých hloubek, můžeme algoritmus SPIHT oproti originálu v [6] modifikovat a tuto vazbu vůbec neuvažovat. Kořeny stromů pak situujeme přímo do detailních koeficientů poslední úrovně. Koeficienty v *cA* následně zpracujeme, jako by byly všechny bez přímých potomků [7].

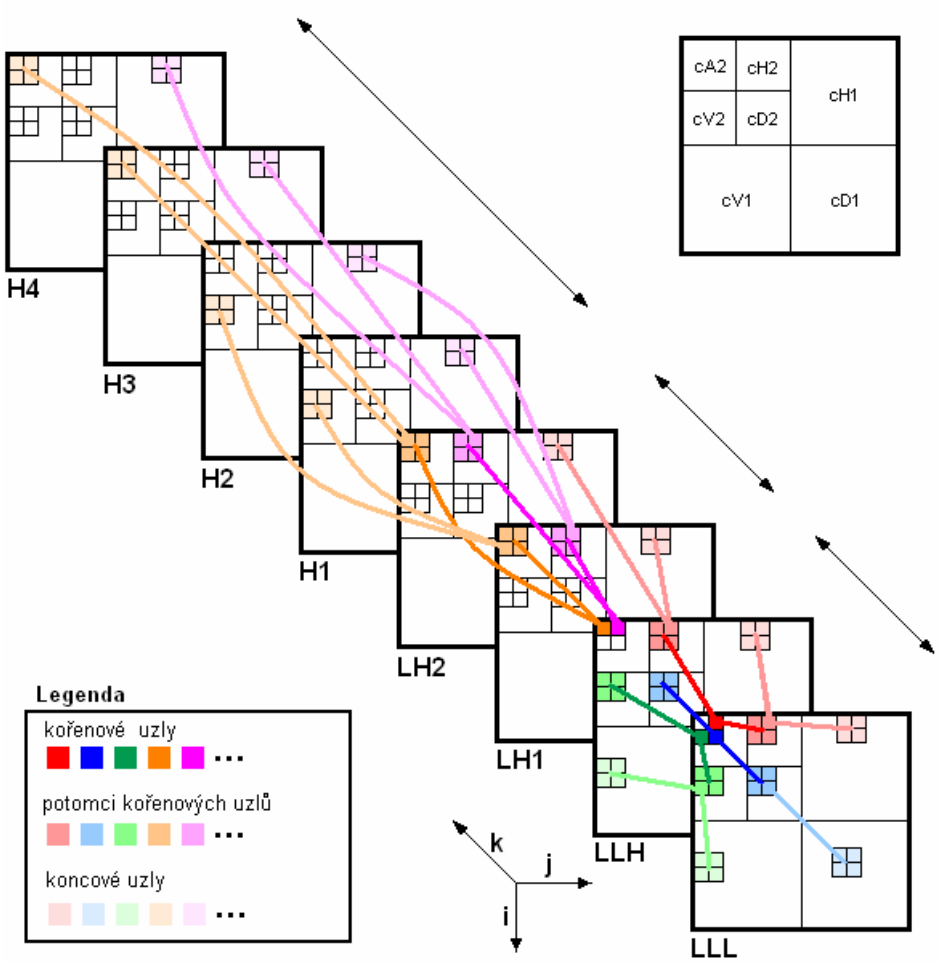


Obr. 4 : Organizace prvků v prostorových stromech

4.4 Organizace dat v časoprostorových stromech

Předchozí popis organizace dat v prostorových stromech popisuje situaci pro statický obraz. Avšak v souvislosti s kódováním videa (sekvence obrázků) je nutné zavést další rozměr, a to je čas. Na *obr. 5* je znázorněn vztah kořenových prvků a jejich potomků pro osm snímků videa, které jsou označovány jako GOF (Group Of Frame).

Rozdíl oproti předchozí organizaci dat je v tom, že každý kořen stromu má osm potomků a opět musí platit, že každý pixel má pouze jediný rodičovský pixel.



Obr. 5 : Organizace prvků v časoprostorových stromech

Podíváme-li se podrobněji na snímek LLL na *obr. 5*, uvidíme v něm skupinu po 2×2 prvcích, jedná se o kořenové prvky. První prvek této skupiny (levý horní bílý) není nikdy uvažován jako kořen stromu. Ostatní tři prvky už mají svoje potomky. Je to proto, aby každý strom začínal pouze jedním koeficientem (kořenem stromu). Každý z těchto prvků má osm potomků. Tito potomci jsou lokalizováni podle *obr. 5*, a to tak,

že v každé nižší úrovni dekompozice je další úroveň prostorového stromu. Tedy, koeficienty v *cA*(2) snímku LLL budou mít potomky v *cH*(2), *cV*(2) a *cD*(2) snímku LLL a LLH. Potom koeficienty v *cH*(2) snímku LLL budou mít potomky v *cH(1*) snímku LLL a LLH a analogicky i *cV*(2) v *cV*(1) a *cD*(2) v *cD*(1) snímku LLL a LLH.

Pro přesnou lokalizaci vzhledem k nadřazeným prvkům je nutné zavést jistá pravidla. Každý prvek, který je součástí stromu a má v dekompozičním obrazci souřadnice *(i,j,k)*, kde souřadnice *i,j* jsou prostorové a *k* je časová souřadnice, platí:
Prvek *(i,j,k)*, bude mít následovníky na souřadnicích *(2i,2j,2k), (2i+1,2j,2k), (2i,2j+1,2k), (2i+1,2j+1,2k), (2i,2j,2k+1), (2i+1,2j,2k+1), (2i,2j+1,2k+1), (2i+1,2j+1,2k+1).*

Pokud má prvek souřadnici $k > 3$, potom nemá žádné potomky a jedná se o koncový prvek stromu.

Toto pravidlo je v celém algoritmu univerzální a platí pro všechny koeficienty kromě těch, které nejsou kořeny prostorového stromu.

Pro naši implementaci algoritmu 3D SPIHT koeficienty *cA* v prvním a druhém snímku následně zpracujeme, jako by byly všechny bez přímých potomků.

4.5 Algoritmus SPIHT

A. Said and W. Pearlman v [5] představili SPIHT (Set Partitioning in Hierarchical Trees) jako vylepšenou verzi algoritmu EZW, který původně navrhl J. M. Shapiro [6]. Hlavní rozdíl je v použití mírně odlišné stromové struktury, která umožňuje častější kódování celé větve stromu jedinou nulou. SPIHT je progresivní iterativní algoritmus schopný i bezeztrátové komprese.

Po rozložení signálu na jednotlivá pásma vlnkovou transformací dochází ke kódování, jehož výstupem je tok bitů, který lze v kterékoliv chvíli přerušit a dosáhnout tak přesně stanovené průměrné délky slova (případně předem daného kompresního poměru). Koeficienty v jednotlivých pásmech vlnkové transformace spolu souvisí a jejich vzájemnou příbuznost lze vyjádřit pomocí stromové struktury. Ukázka stromové struktury příbuznosti koeficientů vlnkové transformace je např. na *obr. 4*.

4.5.1 Průběh algoritmu SPIHT

Dále jsou popsány jednotlivé kroky algoritmu pro jeden prahovací krok. Skládá se ze dvou základních částí: *řadícího průběhu* (sorting pass – vytvoření mapy signifikantních koeficientů) a *upřesňovacího průběhu* (refinement pass – přidání nakonec bitového toku další MSB každého prvku, který byl přidán v minulém srovnávacím průběhu). Při prvním prahovacím kroku se tak provede pouze srovnávací průběh.

Krok algoritmu (každá další prahovací úroveň) je označen celočíselnou proměnnou *n*. Prahovací úroveň se vypočítá jako 2^n, v dalším kroku se *n* sníží o 1.

Pro implementaci algoritmu je potřeba tří typů zásobníků: LSP (*List of Significant Pixels* - seznam signifikantních koeficientů), LIP (*List of Insignificant Pixels* – seznam insignifikantních koeficientů) a LIS (*List of Insignificant Sets* – seznam insignifikantních skupin). Tyto seznamy obsahují souřadnice prvků v dekompozičním obrazci, LIS obsahuje navíc i položku, označující typ skupiny. Rozlišujeme **typ A** – potom hovoříme o všech potomcích daného prvku (specifikovaného souřadnicemi) ve stromu, a **typ B** – potom hovoříme o všech potomcích prvku kromě čtyř bezprostředně následujících.

Na konci každého srovnávacího průběhu obsahuje LSP souřadnice všech signifikantních prvků vzhledem k prahovací úrovni (a to včetně prvků, které byly přidány v předchozích krocích algoritmu).

Označme matici ve které je uložený dekompoziční obrazec $\boldsymbol{C_{i,j}}$. Potom můžeme algoritmus rozepsat do následujících kroků:

1. **Inicializace**

   Nejprve je vypočítána velikost kroku *n* algoritmu jako

$$n = floor\left(\log 2\left(\max\left(|C|\right)\right)\right), \tag{4.1}$$

kde výstupem funkce *floor(x)* je celočíselná část jejího parametru *x* a funkce *max(X)* vrací největší hodnotu prvku matice *X*.

Dále vytvoříme prázdný seznam LSP, LIS inicializujeme tak, aby obsahoval všechny souřadnice prvků z *cH*, *cV* a *cD* poslední úrovně dekompozice (a tyto prvky budou typu A). Seznam LIP bude obsahovat souřadnice prvků z *cA*, *cH*, *cV* a *cD*.

2. **Srovnávací průběh**

Každý prvek z LIP je testován na signifikanci vzhledem k prahovací úrovni (2^n). Pokud je signifikantní, výstupem je 1 následovaná znaménkovým bitem prvku (0 při záporné a 1 při kladné hodnotě prvku) a prvek je přesunut z LIP do LSP. Pokud není signifikantní, výstupem je 0.

Dále je procházen seznam LIS. Nejdříve se zjistí, zda jde o položku typu A nebo položku typu B. Při položce typu A se prochází celým stromem od kořenu daného souřadnicemi z LIS a testují se na signifikanci. Pokud strom neobsahuje signifikantní koeficienty, výstupem je 0 a zpracováváme další prvek LIS. Pokud strom obsahuje alespoň jeden signifikantní koeficient, výstupem je 1 a testujeme každý z bezprostředních následovníků na signifikanci. Pokud je přímý následovník signifikantní, výstupem je 1 následován znaménkovým bitem a prvek je přidán do LSP. Pokud není, výstupem je 0 a pixel je přesunut do LIP. Pokud prvek má více následovníků, přidají se jeho původní souřadnice do LIS jako prvek typu B.

Původní prvek (pokud v něm byl nalezen alespoň jeden signifikantní koeficient) je poté z LIS vyřazen.

Pokud je položka v LIS typu B, pak jsou na signifikanci testovány jeho potomci kromě všech přímých. Pokud alespoň jeden z těchto prvků je signifikantní, je vyřazen z LIS a jeho přímí následovníci jsou přidáni do LIS jako prvky typu A.

3. **Upřesňovací průběh**

Na výstup je vložen *n*-tý MSB hodnoty každého prvku v LSP, kromě těch, které byly přidány v posledním srovnávacím průběhu.

4. **Další krok**

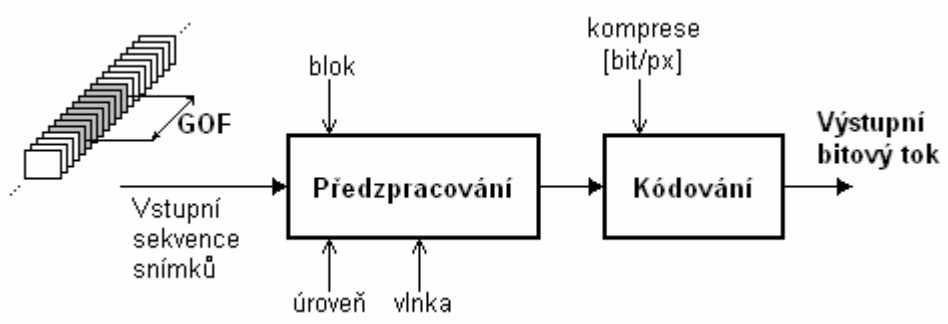
Hodnota *n* je snížena o 1 a proces kódování se opakuje od kroku 2.

Při srovnávacím průběhu jsou vždy procházeny všechny prvky z LIS a LIP, zatímco LSP obsahuje postupně se rozrůstající seznam prvků určených k upřesnění.

Dekodér je koncipován tak, aby pracoval přesně opačně. Protože je cesta kodéru velmi přesně vymezena, je třeba začít dekódovat tok bitů přesně ze stejného místa, kde začalo kódovaní. Díky vlastnosti progresivního kódování je možné proces dekódování kdekoliv přerušit, výsledkem je obrázek kvalitou odpovídající počtu průchodů algoritmu. Posledním krokem po dekódování dekompozičního obrazce je inverzní dvourozměrná vlnková transformace, poté je již získán výsledek ve formě obrazových dat [7].

5 IMPLEMENTACE ALGORITMU V PROGRAMOVÉM PROSTŘEDÍ MATLAB

Navržené kódovací schéma zpracovává vstupní sekvenci snímků po skupinách, která je nazývána GOF (Group Of Frame – skupina snímků). GOF v naší implementaci obsahuje osm snímků. V každém kroku kódování videa je tedy zpracováváno najednou osm snímku, v dalším kroku je zpracovávána další osmice snímků, atd. Celý postup komprese sekvence snímků lze rozdělit do dvou fází: ***předzpracování*** a samotné ***kódování***. V první fázi jsou předzpracovávány vstupní snímky a výstupem jsou vhodně připravená data pro kodér, které jsou v kódovací části zakódovány do výstupního bitového toku.



Obr. 6 : Blokové znázornění navrženého kódovacího schématu

5.1 Předzpracování

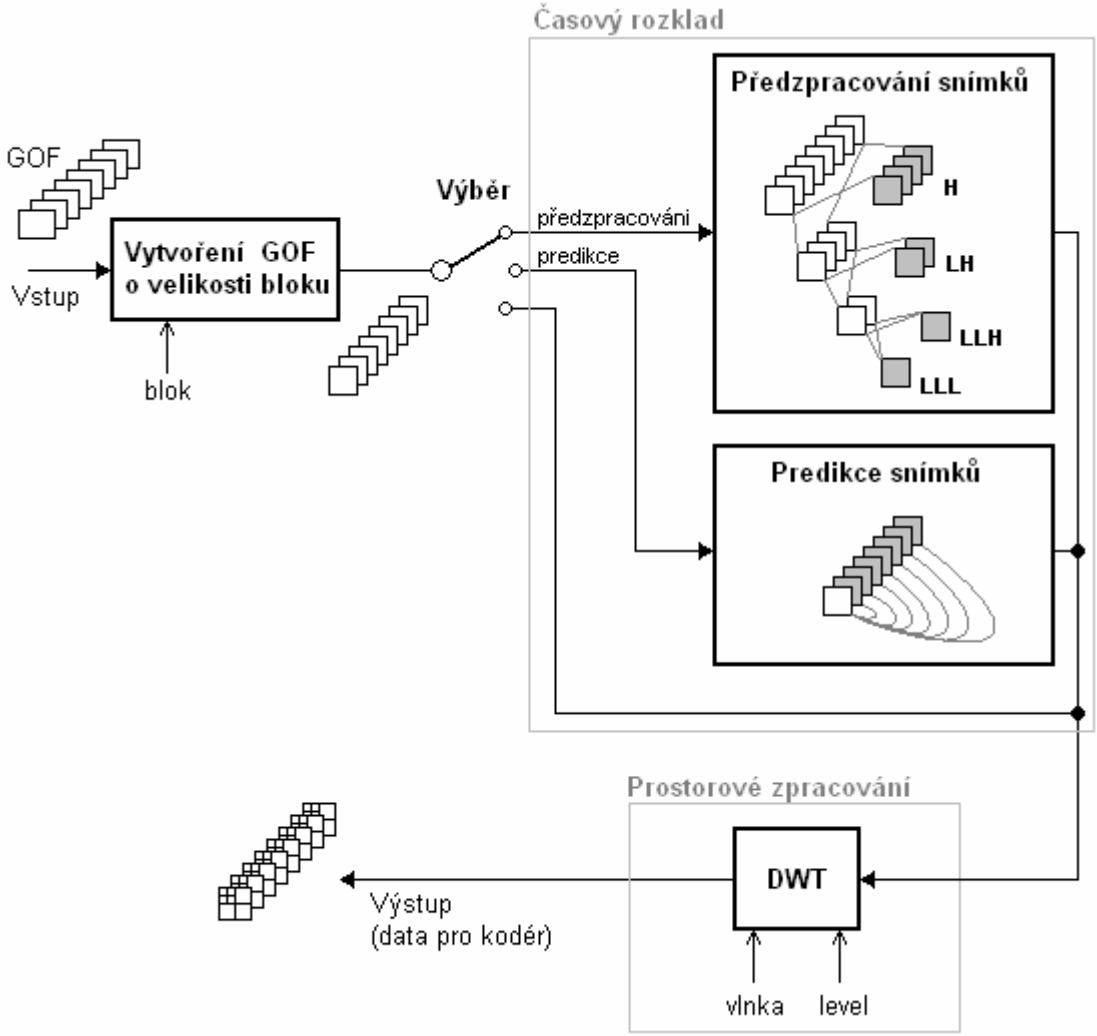
Hlavním úkolem této fáze zpracování je vytvořit vhodné vstupní data pro 3D SPIHT kodér. Jak již bylo řečeno výše, vstupní sekvence je zpracovávána po osmici snímků – GOF. *Obr. 7* ukazuje detailně jednotlivé kroky postupu při předzpracování.

Vstupem je skupina snímků GOF, která je upravena podle zadaného parametru *blok* tak, že je vytvořena nová skupina GOF, kde každý snímek má rozměry (*blok*×*blok*). Jednotkou parametru *blok* je pixel. Toto je realizováno prostým výřezem obrazu podle zadané velikosti bloku. Tato úprava je prováděna pro zjednodušení implementace algoritmu 3D SPIHT.

Následuje část předzpracování, která je nazývána ***časový rozklad***, kde podle uživatelem zvoleném výběru jsou snímky v GOF upraveny. Hlavním cílem je zmenšení časové redundance snímků v GOF. Dále následuje ***prostorové zpracování*** jednotlivých snímků. Zde je realizovaná dvourozměrná diskrétní vlnková transformace s diskrétním časem. Výstupem je potom předzpracovaná skupina snímků GOF, která je připravena k zakódování pomocí 3D SPIHT algoritmu.

5.1.1 Časový rozklad

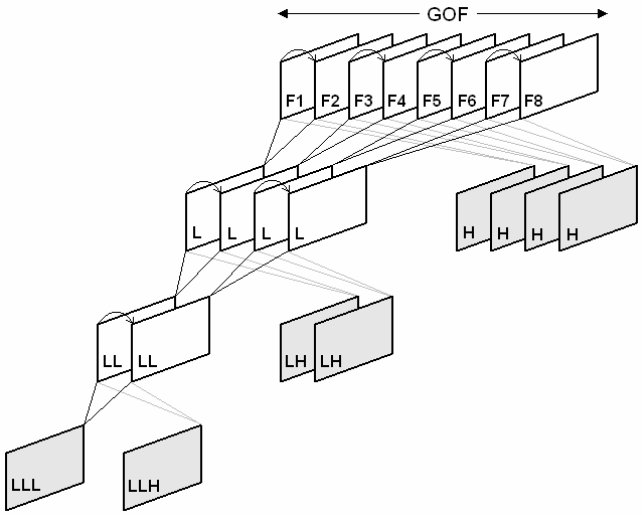
Další krok předzpracování umožňuje vybrat způsob, jak snímky upravit pro zvýšení účinnosti komprese při kódování navrženým algoritmem. Na výběr jsou tři varianty. Buď snímky nijak neupravovat nebo použít navržené schéma podle *obr. 8* nebo použít predikci.



Obr. 7 : Blokové znázornění fáze předzpracování

Předzpracování snímku podle schématu

Obr. 8 naznačuje, jak jsou snímky zpracovávány při výběru *předzpracování snímků*. Úkolem tohoto navrženého schématu, je snížení časové redundance ve skupině po sobě jdoucích snímků. Čím méně bude pohyblivá scéna, tím bude tato metoda účinnější.



Obr. 8 : Předzpracování snímků v GOF

Šedé snímky na *obr. 8* tvoří výstup určený pro další zpracování. Jsou získány podle následujícího pravidla:

$$\begin{aligned} H1 &= F1 - F2 \\ H2 &= F3 - F4 \\ H3 &= F5 - F6 \\ H4 &= F7 - F8 \\ LH1 &= F1 - F3 \\ LH1 &= F5 - F8 \\ LLH &= F1 - F5 \\ LLH &= F1. \end{aligned} \qquad (5.1)$$

Předzpracování s predikcí

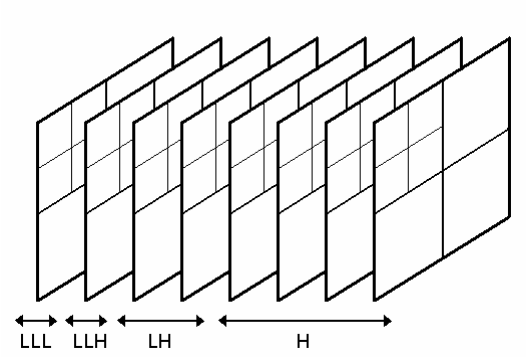
Druhým možným řešením jak snížit časovou redundanci v sekvenci snímků, je zvolit jeden snímek jako referenční a zbylé jako jejich rozdíl s referenčním snímkem:

$$\begin{aligned} M1 &= F1 \\ Mi &= F1 - Fi \quad , \\ &\text{kde} \quad i = 2,3,..8. \end{aligned} \qquad (5.2)$$

5.1.2 Prostorové zpracování

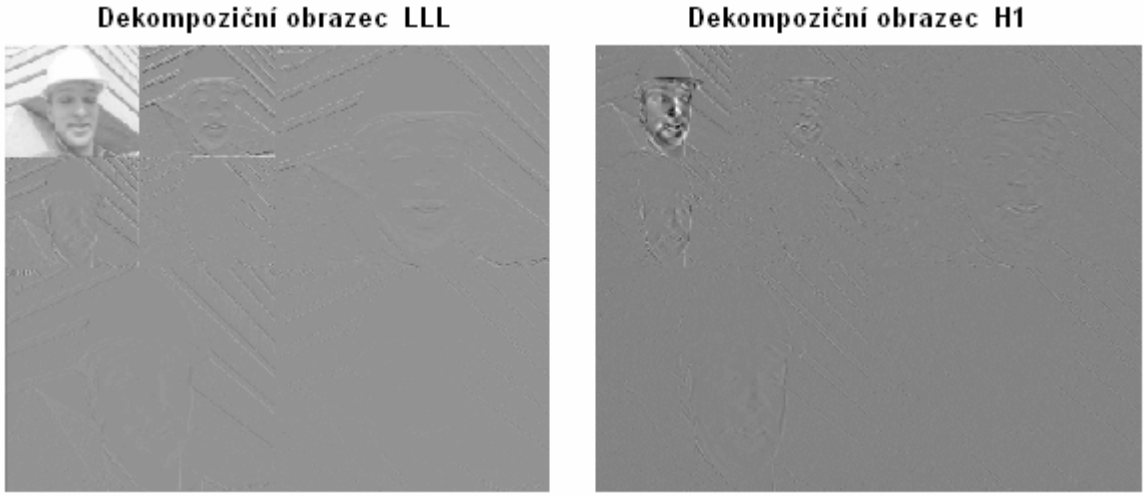
V části *prostorového zpracování* je na každý snímek aplikována dvourozměrná diskrétní vlnková transformace. Volba vlnky a hloubka rozkladu je volena podle potřeby.

Na *obr. 9* je ukázáno celkové uspořádání snímků po předzpracování, které tvoří vstupní datový soubor pro 3D SPIHT kodér.



Obr. 9 : Uspořádání snímků v GOF po DWT (úroveň =2)

Na *obr. 10* je ukázka dvojice reálných snímků z GOF po předzpracování s použitím schématu podle *obr. 8*. Jak je vidět na těchto obrázcích, nejvíce energie je soustředěno do snímku *LLL*, oproti snímkům *H*, ve kterých jsou uchovány detaily (rozdíly snímků). Z toho vyplývá, že pokud nebude v obraze na snímcích v GOF téměř žádný pohyb (statický záběr na neměnnou scénu), nebudou mít snímky *H* téměř žádnou energii.

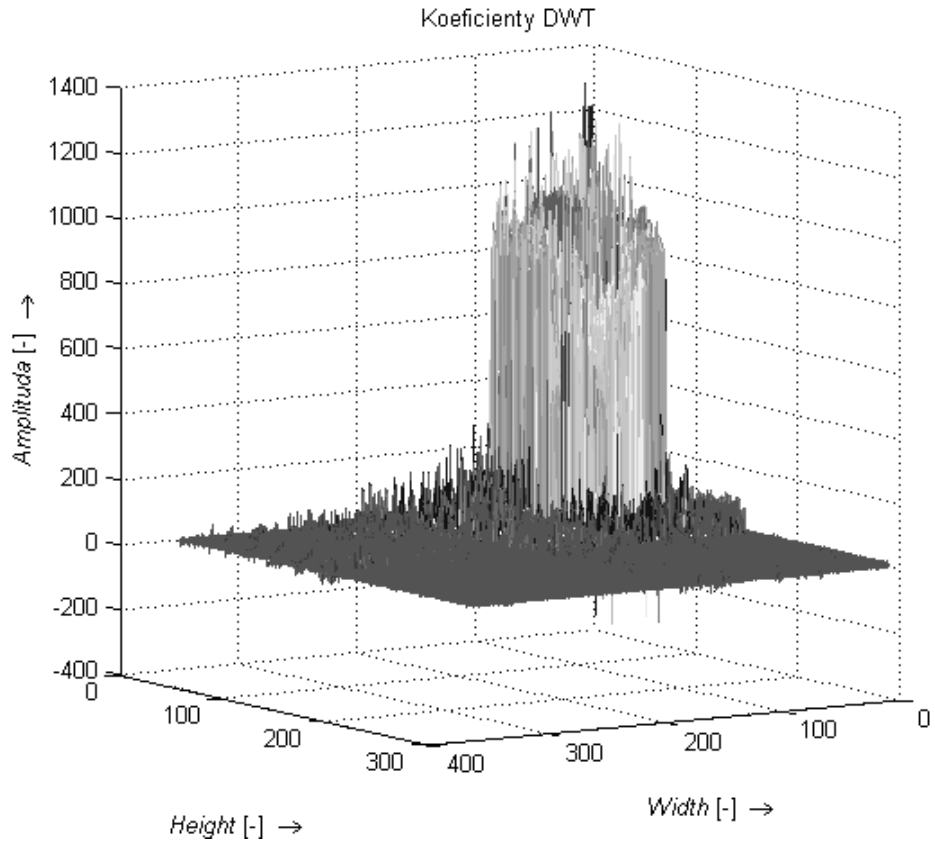


Obr. 10 : Ukázka snímků nejvyšší a nejnižší úrovně po DWT2 v GOF (level = 2, vlnka = „haar“)

Implementace DWT

Pro rozklad obrazového signálu bylo využito nástrojů programového prostředí MATLAB, který nabízí velkou škálu předefinovaných rozkladových filtrů.

Pro představu o běžných velikostech vlnkových koeficientů je přiložen *obr.11*, který znázorňuje 3D zobrazení dekompozičního obrazce (vygenerováno s využitím funkce *mesh* programu MATLAB, jde o dekompozici druhé úrovně obrázku o rozměrech 352×288 px). Z obrázku je patrno, že koeficienty jsou i záporné a že koeficienty v nejvyšší úrovni dekompozice přesahují hodnoty nadřazených pásem i několikanásobně.



Obr. 11 : 3D zobrazení koeficientů dekompozičního obrazce

Pro urychlení celého výpočtu rozkladu se používají různé algoritmy, např. ***Fast Lifting Wavelet Transform,*** který je použit ve standardu JPEG2000. Princip algoritmu spočívá v rozdělení původních filtrů na více menších úseků, tím se dosáhne zmenšení výpočetní náročnosti a snížení paměťových nároků programu [7].

Rychlá implementace diskrétní biortogonální CDF 9/7 vlnkové transformace, která je použitá i v této práci, je získaná z portálu „MATLAB Central“ [22]. V práci je označována jako *„cdf97“*.

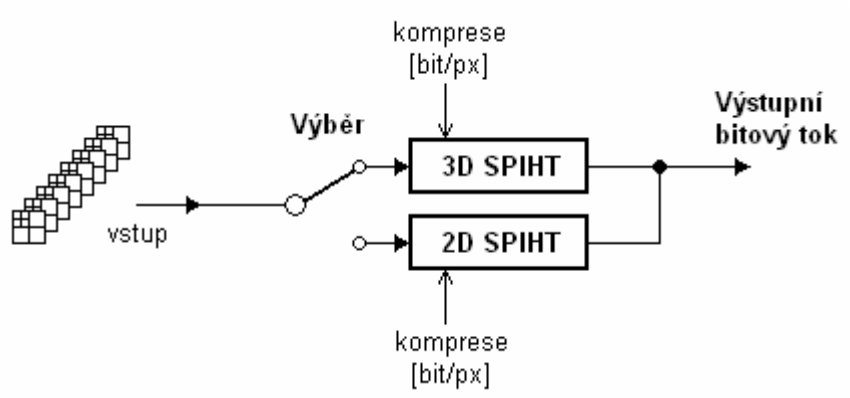
5.2 Kódování

Hlavní funkcí této části je vytvoření výstupního bitového toku, který v sobě nese zakódovanou informaci o snímcích GOF.

Pro možnost porovnání účinnost komprese navrženého 3D SPIHT algoritmu s algoritmem SPIHT, který byl původně navržený pro kompresi statických obrazů, bylo kódovací schéma doplněno o tuto volbu.

Kodér 2D SPIHT realizuje kódování tak, že jednotlivé snímky zpracovává samostatně. Neřeší tedy vůbec redundanci časovou, ale pouze prostorovou.

Parametr *komprese* je potřeba chápat jako vstupní hodnotu pro SPIHT kodér, která udává kolik bitů má mít výstupní bitový tok, nesoucí zakódovaný GOF.



Obr. 12 : Blokové znázornění fáze kódování

5.2.1 Algoritmus 3D SPIHT

Navržený 3D SPIHT algoritmus vychází z algoritmu SPIHT, který je určen pro kódování statických obrazových dat. Označení 3D, lze chápat jako *2D + t*, přibyl tedy třetí rozměr a tím je čas. Toto rozšíření algoritmu je za účelem jeho aplikování na kódování videa (sekvence snímků).

Algoritmus 3D SPIHT opět využívá následující tři seznamy:

- LIP: Seznam nedůležitých koeficientů (List of Insignificant Pixels), obsahuje jednotlivé nedůležité koeficienty.
- LIS: Seznam nedůležitých množin (List of Insignificant Sets), obsahuje stromy nedůležitých koeficientů.
- LSP: Seznam důležitých koeficientů (List of Significant Pixels), obsahuje jednotlivé důležité koeficienty.

Stromy koeficientů mohou být dvojího typu:

- Typ A: zjišťuje se důležitost všech následovníků kořenového (výchozího) prvku.
- Typ B: zjišťuje se důležitost všech následovníků kořenového prvku kromě přímých potomků.

Stejně jako u původního algoritmu SPIHT má i toto rozšíření dvě základní části:

- ***Řadící průchod*** *(Sorting pass)*, během kterého jsou zkontrolovány záznamy v seznamech LIP a LIS a důležité koeficienty jsou přesunuty do seznamu LSP. Pokud je důležitý koeficient součástí stromu, je aplikováno pravidlo na rozdělení množiny a koeficient se stává kořenovým prvkem nového stromu. Jelikož kořenový prvek stromu bývá většinou důležitý, je nezávisle otestována jeho důležitost.

- ***Upřesňující průběh*** *(Refinement pass)*, při kterém je pro zvýšení přesnosti koeficientů ze seznamu LIP poslán na výstup následující bit z binární reprezentace jejich hodnot.

Průběh algoritmu 3D SPIHT

Označme matici, ve které jsou uloženy dekompoziční obrazce, $\boldsymbol{C_{i,j,k}}$. Potom můžeme algoritmus rozepsat do následujících kroků:

1. **Inicializace**

Nejprve je vypočítána velikost kroku *n* algoritmu jako

$$n = floor\left(\log 2\left(\max\left(\left|C\right|\right)\right)\right), \tag{5.3}$$

kde výstupem funkce *floor(x)* je celočíselná část jejího parametru *x* a funkce *max(X)* vrací největší hodnotu prvku matice *X*, která je v tomto případě trojrozměrná.
Hodnota prahu je stanovena jako 2^n.

Dále vytvoříme prázdný seznam LSP. LIS inicializujeme tak, aby obsahoval všechny souřadnice prvků z *cH*, *cV* a *cD* poslední úrovně dekompozice prvního a druhého snímku a *cA*, *cH*, *cV* a *cD* poslední úrovně dekompozice třetího a čtvrtého snímku. Všechny prvky v seznamu LIS jsou označeny jako typ A.

Seznam LIP bude obsahovat souřadnice prvků z *cA*, *cH*, *cV* a *cD* poslední úrovně dekompozice prvního, druhého, třetího a čtvrtého snímku. Toto je dáno právě časoprostorovou strukturou stromu, která je zobrazena např. na *obr. 5*.

2. Srovnávací průběh

Tato fáze algoritmu je již stejná jako u klasického SPIHT, proto je vysvětlena jen v bodech:

- Otestuje se, zda-li jsou koeficienty v LIP důležité (vzhledem prahu 2^n):
  ANO: koeficient je důležitý, na výstup se pošle 1 následována znaménkovým bitem a koeficient je přesunut do LSP.
  NE: koeficient není důležitý, na výstup se pošle 0.

- Otestuje se důležitost všech stromů v LIS.
  Strom je důležitý, pokud je některý z následovníků důležitý (testují se následovníci dle typu stromu – A/B):

  Typ A:
  ANO: strom je důležitý: na výstup se pošle 1 a přímí potomci jsou otestováni obdobně jako v předchozím bodě:
  ANO: přímý potomek je důležitý, na výstup se pošle 1 a je přidán do LSP,
  NE: přímý potomek není důležitý, na výstup se pošle 0 a je přidán na konec LIP.
  Pokud přímí potomci mají další následovníky:
  ANO: přesunutí stromu na konec LIS jako typ B,
  NE: odstranění stromu z LIS,
  NE: strom není důležitý, na výstup se pošle 0.

  Typ B:
  ANO: strom je důležitý, na výstup se pošle 1 a všichni přímí potomci jsou přidáni na konec LIS jako typ A, rodičovský strom se odstraní z LIS,
  NE: strom není důležitý, na výstup se pošle 0.

3. Upřesňující průběh

Na výstup je vložen *n*-tý MSB hodnoty každého prvku v LSP, kromě těch, které byly přidány v posledním srovnávacím průběhu.

4. Další krok

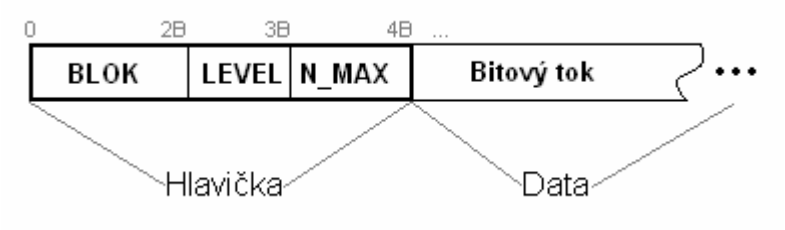
Hodnota *n* je snížena o 1 a proces kódování se opakuje od kroku 2 pro novou hodnotu prahu.

Při srovnávacím průběhu jsou vždy procházeny všechny prvky z LIS a LIP, zatímco LSP obsahuje postupně se rozrůstající seznam prvků určených k upřesnění.

Dekodér je opět koncipován tak, aby pracoval přesně opačně. Protože je cesta kodéru velmi přesně vymezená, je třeba začít dekódovat tok bitů přesně ze stejného místa, kde začalo kódovaní.

5.2.2 Výstupní zakódovaná informace

Pro účely demonstrace schopnosti 3D SPIHT kodéru postupně dekódovat bitový tok, bylo nutné vhodně tyto data ukládat. Na *obr. 13* je znázorněna struktura výstupního bitového toku, který se skládá z ***hlavičky*** a ***datové části***. Datová část obsahuje výstupní bity 3D SPIHT kodéru při kódování jednoho GOF. Chceme-li však demonstrovat pouze princip algoritmu, je tato struktura postačující. Pokud by měl být ale vytvořen soubor, který v sobě nese zakódovanou celou vstupní sekvenci, musela by hlavička obsahovat také informace o použité vlnce, počtu GOF, délce datové části a jiné potřebné informace pro úspěšné dekódování.



Obr. 13 : Struktura výstupního bitového toku

Hlavička:

- LEVEL - úroveň prostorové vlnkové dekompozice
- BLOK - velikost kódovaného bloku dat
- N_MAX - maximální velikost bitové hladiny

Data:

- Bitový tok - jednotlivé výstupní bity 3D SPIHT kodéru

5.3 Vstupní testovací sekvence snímků

Prostorové rozlišení snímků je 352×288 pixelů, což odpovídá formátu CIF. Barevný formát je YUV 4:2:0. *Y* je jasová (luminanční) složka a *U* a *V* jsou barvonosné (chrominanční) složky. Barevná hloubka je 8 bitů na pixel pro každou složku. Navzorkování v poměru 4:2:0 znamená, že barevné složky vůči jasové jsou v tomto případě v poměr 4:1 a barevná složka tedy obsahuje čtvrtinu bodů vůči jasové, nebo-li jinak řečeno na čtyři jasové body připadá pouze jeden barevný.

Obr. 14 : Ukázky snímků testovacích sekvencí
(Postupně zleva doprava: sekvence „foreman_cif.yuv“, sekvence „akiyo_cif.yuv“, sekvence „hall_monitor_cif.yuv“, sekvence „coastguard_cif.yuv“)

Sekvence *„foreman_cif.yuv“*

Kamera držená v ruce (mírné chvění obrazu) snímá předáka na stavbě, který výrazně artikuluje. Poté, co ukáže na rozestavěnou budovu, se záběr přesouvá na ni.

Sekvence *„akiyo_cif.yuv“*

Statická kamera snímá moderátorku, která čte zprávy. Jediný pohyb ve scéně je její mrkání očí a mírné naklánění hlavy při čtení.

Sekvence *„hall_monitor_cif.yuv“*

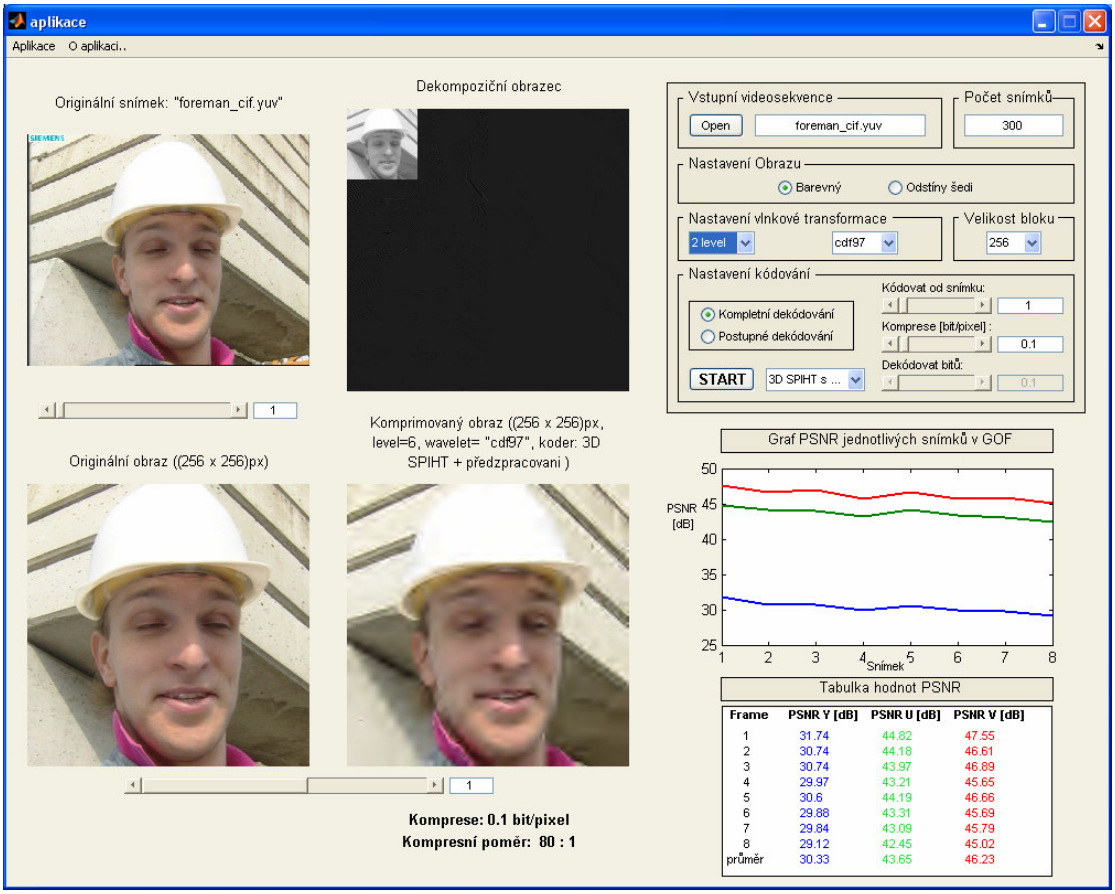
Záběr bezpečnostní kamery namířené na chodbu, po které projdou dva muži.

Sekvence *„coastguard_cif.yuv“*

Záběr kamery pohybující se současně s plující lodí pobřežní policie. Scéna obsahuje velkou vodní plochou, která je charakteristická svou měnící se strukturou.

5.4 Aplikace

Pro lepší přístup k programu bylo vytvořeno grafické uživatelské rozhraní, které usnadní uživateli ovládání celého programu. Aplikace se spustí po zadání v příkazovém poli „*Command Window*“ příkazu „***aplikace3DSPIHT*“** (navíc je nutné mít v „*Current directory*“ nastavenou cestu ke zdrojovým souborům této aplikace). Aplikace byla testována na verzi MATLAB 7.2.0.232(R2006a).



Obr. 15 : Ukázka aplikace

5.4.1 Popis aplikace

Aplikace se skládá ze tří základních částí, levá část slouží k **vizualizaci** vstupních a výstupních obrazových dat, v pravém horním rohu se nachází **ovládání** a pod ním je vyhrazeno místo na zobrazení **výsledků měření PSNR**, určující objektivní kvalitu komprimovaných dat, v podobě grafu a příslušné tabulky.

Jako první krok po spuštění aplikace je potřeba vybrat vstupní videoskvenci (musí být formátu jaký je uveden v *kapitole 5.3*), na které chceme testovat algoritmus 3D SPIHT. To provedeme kliknutím na tlačítko *Open*, otevře se klasické okno pro výběr souboru ve formátu **.yuv*. Po potvrzení vybrané videosekvence se

ve vedlejších polích objeví její název a celkový počet snímků obsažených ve vybraném souboru. V prvním okně se zobrazí její první snímek. Načtenou sekvenci je možné procházet pomocí posuvníku umístěného pod náhledem. Ve vedlejším okně je vždy zobrazen dekompoziční obrazec aktuálního snímku, který je ovlivněn parametry vlnkové transformace (level - úroveň dekompozice, vlnka). Při každé změně jakéhokoliv parametru vlnkové transformace je okamžitě aktualizován zobrazovaný dekompoziční obrazec. Další parametr je nastavení *Velikost bloku*, který definuje, jaké rozměry obrazu budou použity pro kódování. Navržený algoritmus umí pracovat pouze s bloky čtvercového tvaru.

V položce *Nastavení obrazu*, lze pomocí přepínače zadat, zda má být počítáno pouze s jasovou složkou obrazu, nebo zda se mají kódovat i barvonosné složky.

Poslední skupina ovládacích prvků se týká už samotného *Nastavení kódování*. Pomocí přepínače lze zadat, zda se po spuštění kódování má zakódovaný bitový tok ihned dekódovat. Nebo druhou volbou je, že se bude výsledný bitový tok dekódovat postupně, podle zadaného počtu bitů, kterou uživatel nastaví třetím posuvníkem.

První posuvník umožňuje zadat, jaký snímek bude první v kódované skupině snímků GOF. Druhým posuvníkem se nastavuje velikost komprese, jednotkou je bit/pixel.

Uživatel má možnost vybrat si ze čtyř kodérů, pomocí *List boxu* umístěného vedle tlačítka *START*, které spouští podle zadaných parametrů kódování. Pokud bylo vybráno *Postupné dekódování*, změní se toto tlačítko na *STOP*, které po stisku ukončí tuto fázi simulace.

Po každém dekódování proběhne měření PSNR, které určuje, jak velká je změna rekonstruovaných dat oproti originálním (viz. *kapitola 5.5.2*). Vypočítané hodnoty jsou zobrazeny v tabulce v pravém dolním rohu aplikace a graficky znázorněny. Dále se zobrazí ve třetím náhledovém okně první snímek z GOF a ve čtvrtém jeho odpovídající snímek po rekonstrukci. Důležitými údaji, které se nacházejí pod ním, jsou velikost komprese a kompresní poměr. Pomocí posuvníku umístěného pod těmito náhledy, lze vybrat, který snímek z GOF se má zobrazit.

Aplikaci je možno ukončit pomocí „*křížku*“ v pravém horním rohu nebo vybráním položky *Aplikace/Konec*.

5.5 Způsoby vyhodnocování kvality obrazu

V případě použití ztrátových kompresních technik dochází vlivem kvantování k nevratné ztrátě části původní informace. Tuto ztrátu lze kvantifikovat několika způsoby.

5.5.1 Střední kvadratická chyba - MSE (Mean Square Error)

Jedná se o velmi často používanou metodu popisující změnu komprimovaného obrazu oproti obrazu originálnímu. Není nejvhodnější z hlediska subjektivní kvality obrazu, kdy se i obraz s velkou MSE může jevit stejně kvalitně jako originál.

MSE je dána následujícím vztahem [1]

$$MSE_{[-]} = \frac{1}{M \cdot N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(x(m,n) - \hat{x}(m,n) \right)^2 , \qquad (5.3)$$

kde: $\hat{x}$ jsou obrazové body (pixely) původního obrazu

$\hat{x}(m,n)$ jsou obrazové body rekonstruovaného obrazu

M, N jsou rozměry obrazu.

5.5.2 Špičkový poměr signál - šum - PSNR (Peak Signal to Noise Ratio)

Vychází ze střední kvadratické chyby MSE a je dán vztahem [1]

$$PSNR_{[dB]} = 10 \cdot \log_{10} \left(\frac{255^2}{MSE} \right) . \qquad (5.4)$$

5.5.3 Subjektivní hodnocení

K posouzení kvality se využívá i metod subjektivních. Jejich konkrétní realizace vychází z hodnocení výsledného obrazu statisticky dosti velkou a na sobě vzájemně nezávislou skupinou lidí.

5.6 Testování algoritmu

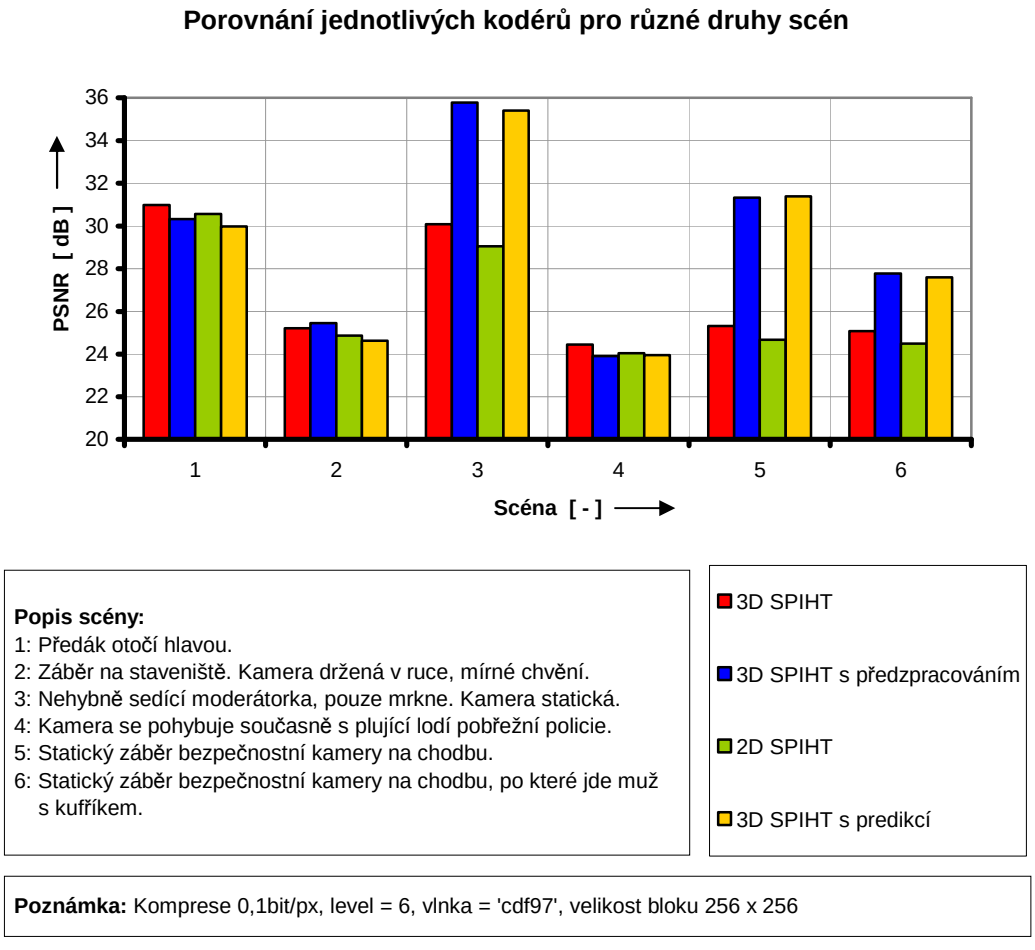
Testování účinnosti navrženého algoritmu je rozděleno do několika částí:

- Testování kodérů
- Úroveň dekompozice
- Druh vlnky
- Velikost komprese
- Mezirámcová účinnost kódování

5.6.1 Testování kodérů

Jak bylo vysvětleno výše, je možno provést testování pro čtyři různé druhy kódovacích schémat:

- 3D SPIHT : Snímky v GOF nejsou nijak předzpracovávány, pouze je na každý aplikována vlnková transformace. Kódování je realizováno algoritmem 3D SPIHT.
- 3D SPIHT s předzpracováním : Snímky v GOF jsou předzpracovány podle *obr. 8* a na každý je aplikována vlnková transformace. Kódování je realizováno algoritmem 3D SPIHT.
- 2D SPIHT : Snímky v GOF nejsou nijak předzpracovávány, pouze je na každý aplikována vlnková transformace. Kódování je realizováno algoritmem SPIHT jednotlivě na každý snímek (implementace algoritmu je převzata z práce [7]).
- 3D SPIHT s predikcí : Snímky v GOF jsou předzpracovány s predikcí a na každý je aplikována vlnková transformace. Kódování je realizováno algoritmem 3D SPIHT.



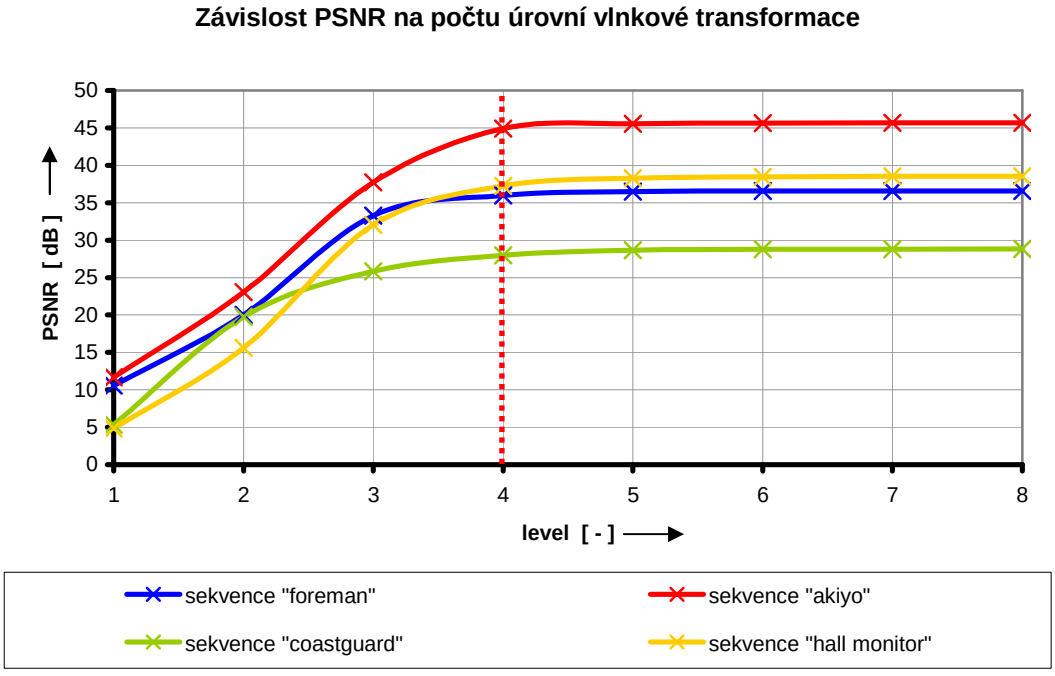
Obr. 16 : Graf závislosti PSNR jednotlivých kodérů pro různé obrazové scény

Testování bylo provedeno pro různé scény, které mají jisté obrazové rysy. Popis jednotlivých scén je součástí naměřeného grafu na *obr. 16*. V *příloze A* jsou potom zobrazeny vždy první snímky testovaných GOF pro různé obrazové scény. Čísla u jednotlivých snímků v příloze odpovídají číslům uvedeným u popisu scény v grafu na *obr 16*.

Jak lze vypozorovat z grafů, největších rozdílů mezi kodéry je dosaženo ve scénách, ve kterých je zaznamenaný minimální pohyb (scéna 3, 5 a 6). Při stejné velikosti komprese dosahují kodéry s předzpracováním mnohem větších hodnot PSNR (průměrná hodnota PSNR snímků v GOF), tudíž rekonstruovaný obraz se více podobá originálnímu, než kodéry bez předzpracování. Předzpracováním je totiž zmenšena časová redundance v po sobě jdoucích snímcích.

5.6.2 Úroveň dekompozice

Jak ukazuje *obr. 17*, hodnota PSNR se už téměř nemění u úrovně dekompozice vlnkové transformace od hodnoty čtyři a více. Toto platí i pro jiné velikosti kódovaných bloků, než je znázorněn na tomto grafu, kde byla velikost bloku 256 pixelů. Ostatní proměřené charakteristiky pro velikosti bloku 128 a 64 pixelů jsou uvedeny v příloze této práce. Opět je na nich patrné, že platí výše uvedené tvrzení.



Poznámka: Komprese 0,5bit/px, vlnka = 'cdf97', velikost bloku 256 x 256,
kodér: 3DSPIHT s předzpracováním

Obr. 17 : Graf závislosti PSNR na počtu úrovní vlnkové transformace pro blok 256 pixelů

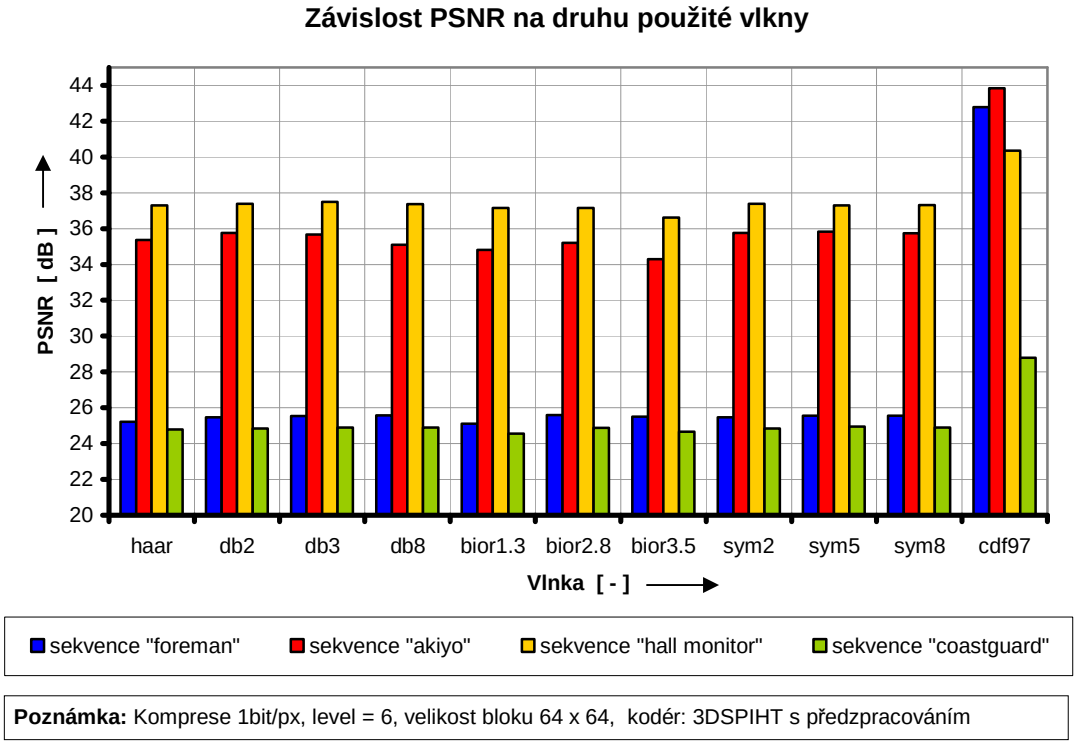
5.6.3 Druh vlnky

V této fázi testování byl zkoumán vliv použité vlnky při vlnkové transformaci. Graf na *obr. 18* ukazuje, že jak bude velký vliv vlnky závisí na charakteru obrazových dat. Rozdíl mezi jednotlivými vlnkami je maximálně 1 dB PSNR. Výjimkou je ale použitá biortogonální *„cdf97“* vlnka, u které je vidět rozdíl až několik jednotek dB. Její hlavní výhodou je, že po transformaci dává mnohem méně vlnkových koeficientů (počet koeficientů je stále stejný, ale hodně jich má nulovou hodnotu). Toto ilustruje *obr. 19*, na kterém je trojrozměrné zobrazení dekompozičního obrazce pro použitou vlnku *„haar“* a *„cdf97“*.

5.6.4 Velikost komprese

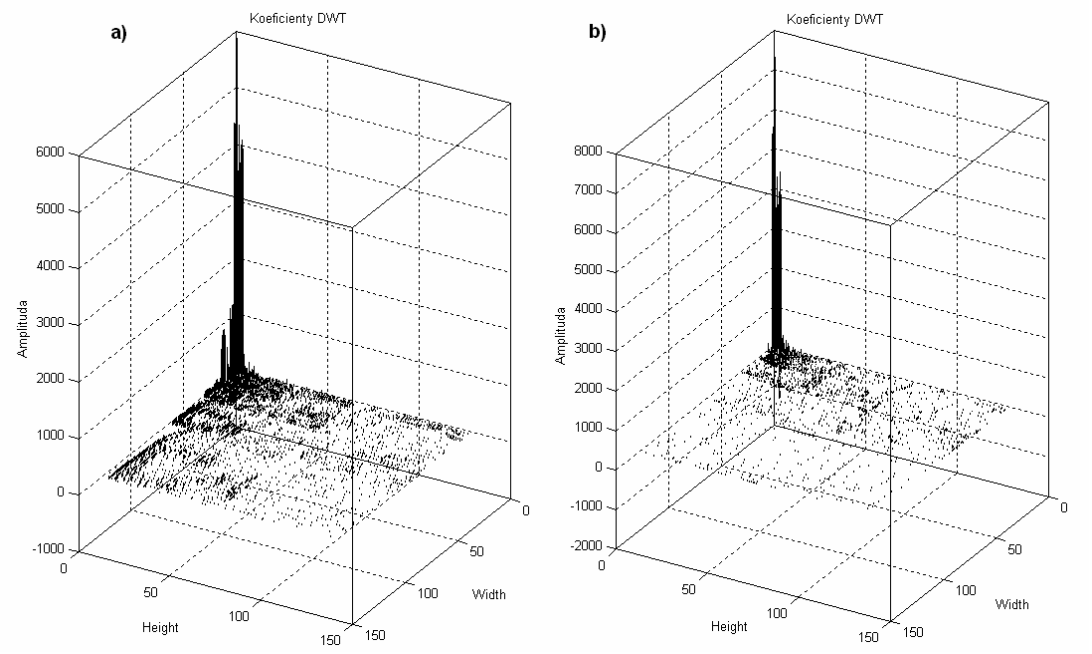
Obr. 20 ukazuje grafickou závislost PSNR na velikosti komprese. Jiným vyjádřením velikosti komprimace může být kompresní poměr, který udává podíl velikosti původních dat ku velikosti komprimovaných dat. Např. hodnotě komprese 0,2 bitů na pixel odpovídá kompresní poměr 40:1. *Obr. 20* ukazuje hodnoty PSNR pro běžné velikosti komprese. Druhý obrázek (*obr. 21*) ukazuje, že přibližně při velikosti komprese vyšší jak 5 bitů na pixel se hodnota PSNR blíží nekonečnu, což jinými slovy znamená, že komprese je bezeztrátová. To ale neplatí pro testovací sekvenci *„coastguard“*, u které se hranice, která popisuje bezeztrátovost, posunula až na hodnotu kolem 7,5 bitů na pixel. To je způsobeno složitou obrazovou strukturou (vlnky na vodě), které při převodu do frekvenční oblasti způsobí, že výsledný dekompoziční obrazec bude obsahovat hodně vysokofrekvenčních koeficientů, proto bude potřeba pro kódování použít více bitů.

Z toho vyplývá, že velikost výsledné komprese závisí na charakteru kódované informace v obraze.



Poznámka: Komprese 1bit/px, level = 6, velikost bloku 64 x 64, kodér: 3DSPIHT s předzpracováním

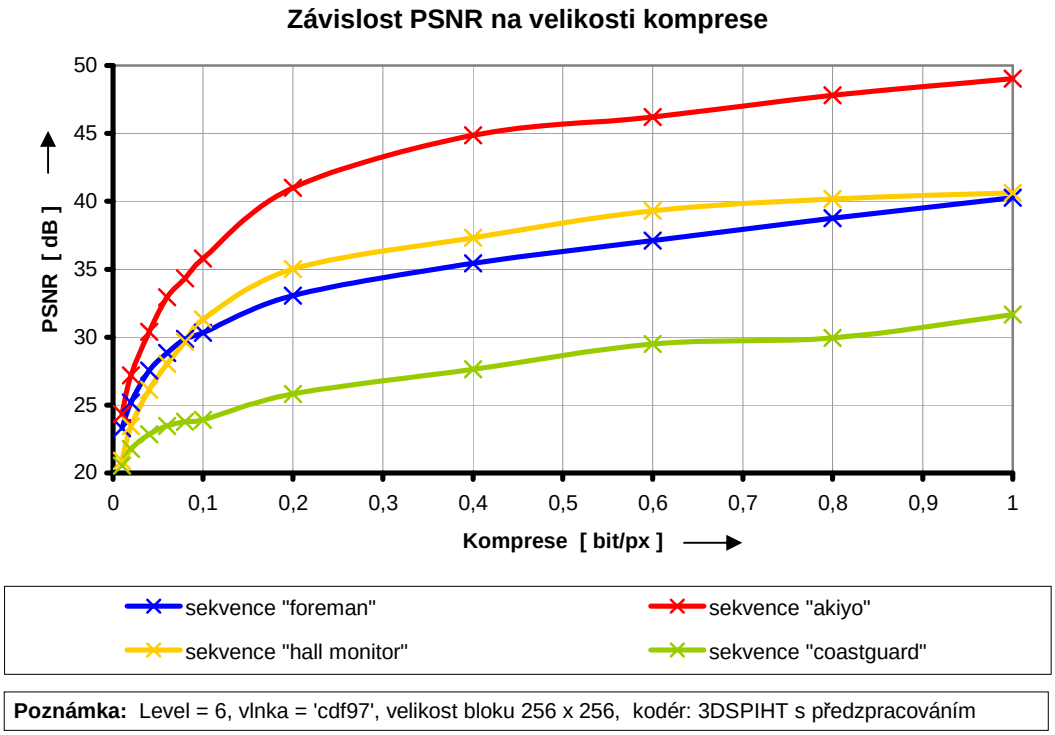
Obr. 18 : Graf závislosti PSNR na druhu použité vlnky



Obr. 19 : Ukázka koeficientů vlnkové transformace (level = 5)

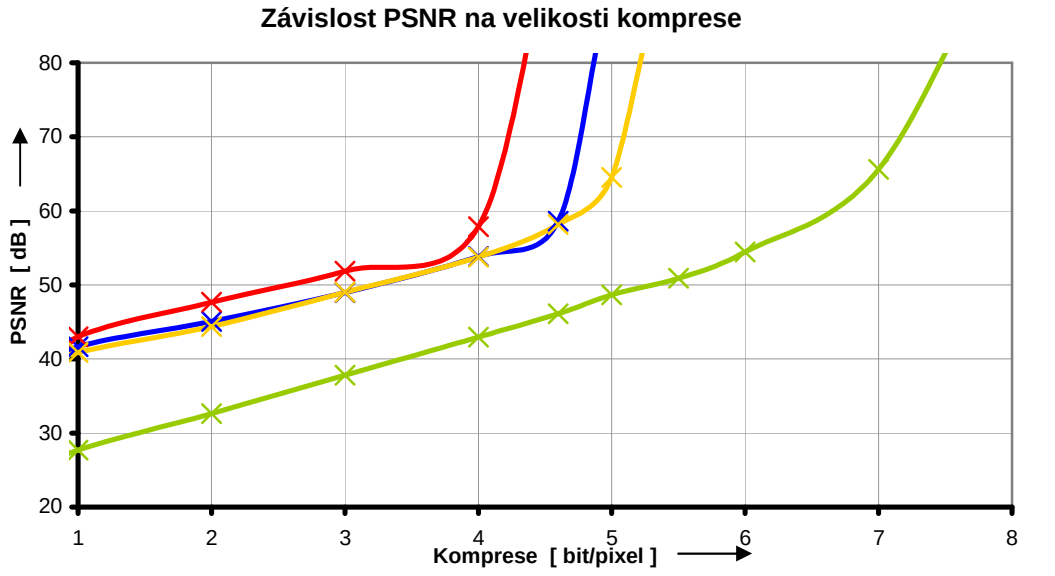
a) vlnka „haar“

b) vlnka „cdf97“



Poznámka: Level = 6, vlnka = 'cdf97', velikost bloku 256 x 256, kodér: 3DSPIHT s předzpracováním

Obr. 20 : Graf závislosti PSNR na velikosti komprese (velikost bloku 256 px)



Poznámka: Level = 4, vlnka = 'cdf97', velikost bloku 32 x 32, kodér: 3DSPIHT s předzpracováním

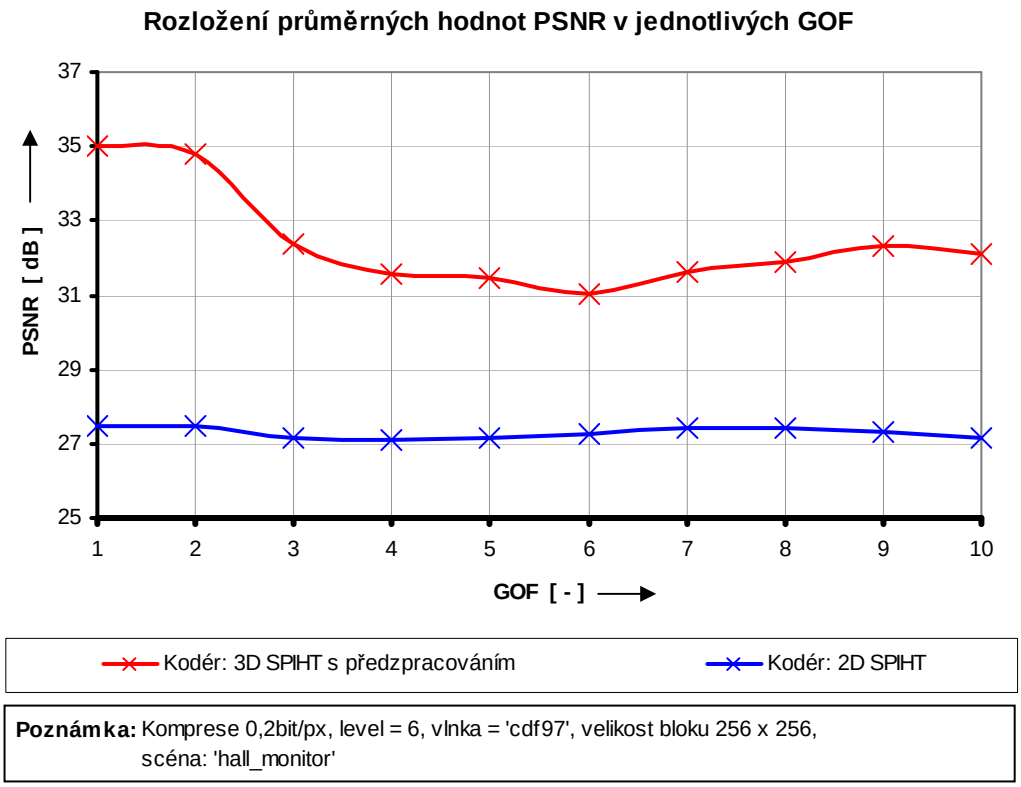
Obr. 21 : Graf závislosti PSNR na velikosti komprese (velikost bloku 32 px)

5.6.5 Mezirámcová účinnost kódování

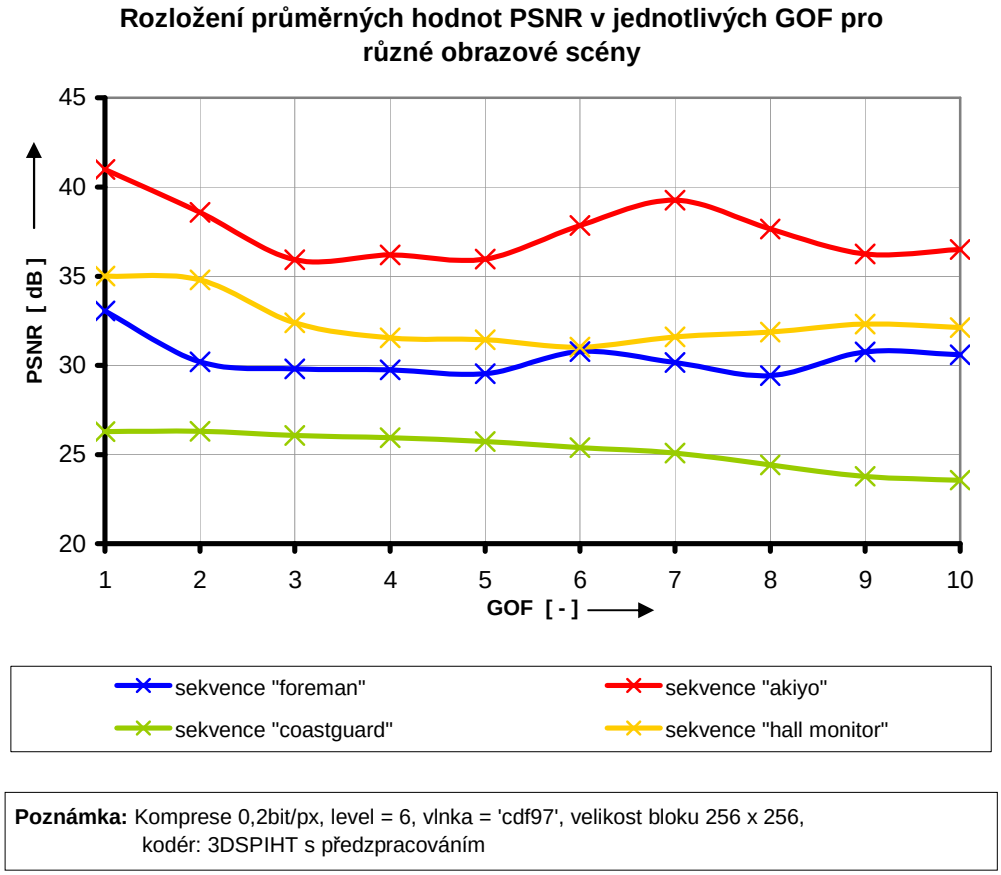
Na *obr. 22* je vidět, jak vypadají průměrné hodnoty PSNR v prvních deseti GOF testovací sekvence *„hall monitor“*, která poskytuje dvě charakteristické obrazové scény, statický záběr chodby a pohybujícího se člověka po chodbě. Porovnávány byly dvě kódovací schémata, navržený 3D SPIHT s předzpracováním a 2D SPIHT kodér. Z grafu jde vidět, že navržený kodér mnohem lépe snižuje časovou redundanci obrazu.

U prvních dvou skupin GOF, kde je pouze záběr kamery na prázdnou chodbu, je rozdíl mezi těmito kodéry až 7 dB. Na dalších GOF, kde se již objevuje ve scéně pohyb, se tento rozdíl snižuje na 4 dB, což je pořád vysoká hodnota při hodnocení kvality obrazu.

Pro představu o průměrných hodnotách PSNR v jednotlivých GOF pro různé druhy scén byl naměřen graf níže (*obr. 23*).



Obr. 22 : Graf závislosti průměrných hodnot PSNR v jednotlivých GOF, scény „hall monitor“



Obr. 23 : Graf závislosti průměrné hodnoty PSNR jednotlivých GOF pro různé obrazové scény

5.6.6 Subjektivní hodnocení kvality komprimovaného obrazu

Jak je vidět na *obr. 24*, i při vysokém kompresním poměru je obrazová informace dobře čitelná, ale hlavním rozdílem oproti kompresním technikám založeným na DCT transformaci (jako je např. JPEG, MPEG2 apod.) je, že se v obraze nenacházejí rušivé blokové artefakty. Vysoká komprese se na snímku kódovaném pomocí DWT a příslušných algoritmů projeví jako rozmazání detailních ploch, což působí pro pozorovatele méně rušivě.

Obr. 24 : Srovnání původního a komprimovaného snímku (3D SPIHT kodérem), komprese 0,1bit/pixel (kompresní poměr 80:1)

5.7 Možné rozšíření navrženého kodéru

Dalším možným rozšířením navrženého kodéru je zavedení vektorové pohybové kompenzace, kdy je sledován pohybující se objekt scény. Ten se zakóduje a v dalších rámcích jsou přenášeny pouze vektory pohybu, které udávají směr pohybu zakódovaného objektu.

Protože výstupem navrženého kodéru je sled „1“ a „0“, je možné pro zvýšení účinnosti komprese aplikovat na výstupní bitový tok ještě nějaký z druhů bezeztrátové komprese (např. RLE nebo aritmetické kódování).

6 ZÁVĚR

Vlnkově založené kódování videosignálu přináší do tohoto vědního směru, zabývajícího se kompresí a kódováním videa, nové možnosti. Jsou vyvíjeny různé metody využívající vlastností vlnkové transformace, které se především zaměřují na kódování vlnkových koeficientů. Výborných výsledků dosahují metody využívající stromových struktur pro kódování koeficientů dekompozice.

Jeden z představitelů tohoto algoritmu je SPIHT (Set Partitioning In Hierarchical Trees) [5, 6], autorů Saida a Pearlmana, který prokázal, že dosahuje výborných výsledků v kódování statických obrazů. Tato práce se zabývá jeho rozšířením (3D-SPIHT) na kódování videosignálu, používající trojrozměrných časoprostorových struktur stromů.

První část práce je věnována teorii, která je převážně zaměřena na vlnkovou transformaci, popis videosignálu, jeho parametrů a vlastností. Dále jsou zde uvedeny základní metody komprese a popis algoritmu SPIHT.

Druhá část práce obsahuje základní implementaci komprese videosignálu využívající vlnkové transformace. Jedná se o kodér snižující prostorovou, ale i časovou redundanci v obraze. Hlavním prvkem je navržený algoritmus 3D SPIHT, který kóduje vstupní sekvenci snímků po skupinách, nazývaných GOF.

Pro účely snadného a intuitivního ovládání kodéru byla vytvořena aplikace. Uživatel má možnost měnit parametry kodéru a sledovat změny na zobrazovaných náhledech snímků a naměřených grafech. Pro testování byly zvoleny čtyři sekvence snímků, které obsahují různé obrazové scény, s různými charakteristickými rysy.

V neposlední řadě jsou zde uvedeny způsoby vyhodnocení kvality obrazu po jeho kompresi.

Pro různé nastavené parametry kodéru byly naměřeny hodnoty, sestrojeny grafy a učiněny patřičné závěry. Navržený kodér 3D SPIHT byl především porovnáván s jeho mateřskou implementací a jak ukazují výsledky testování, zvýšila se jeho účinnost komprese řádově o několika dB, při objektivním hodnocení pomocí PSNR.

Jak již ukázala řada experimentálních testování, nejlepší nástroje současnosti, co do efektivity komprese, jsou založeny na DWT transformaci. Výzkum v této oblasti není zdaleka uzavřený, v budoucnu lze předpokládat zejména zvýšený zájem o oblast vektorového kódování dekompozičního obrazce (jejímž úspěšným představitelem je v současné době právě především metoda SPIHT). Navíc, efektivita komprese jako taková, není rozhodně jediným kritériem, kterému je věnována pozornost. Dalšími důležitými rysy je například odolnost kódovaní proti chybám v přenosovém řetězci, bezpečnost při přenosu, možnost náhodného přístupu do zakódovaného souboru dat, objektový způsob práce a jiné.

LITERATURA

[1] UCHYTIL, S. : Komprese ve videokomunikaci v multimediálních sítích. *Elektrorevue.cz* [online]. 2005 [cit. 2007-12-07]. Dostupný z WWW: <http://www.elektrorevue.cz/clanky/04045>.

[2] BODEČEK, K. : Komprese rozměrných obrazů pomocí JPEG2000. *Elektrorevue.cz* [online]. 2005 [cit. 2007-12-07]. Dostupný z WWW: <http://www.elektrorevue.cz/clanky/05011>.

[3] VOJÁČEK, A. : *Vlnková transformace pro číslicové zpracování signálů* [online]. 14.10.2006 [cit. 2007-12-07]. Dostupný z WWW: <http://automatizace.hw.cz/clanek/2006101401>.

[4] BODEČEK, K., BŘEZINA, M. : Škálovatelná komprese videa. *Elektrorevue.cz* [online]. 2006-02-28 [cit. 2007-12-07]. Dostupný z WWW: <http://www.elektrorevue.cz/clanky/06013/index.html>.

[5] SAID, A., PEARLMAN, W.A. : An Embedded Wavelet Video Coder Using Three-Dimensional Set Partitioning in Hierarchical Trees (SPIHT), IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, 1997.

[6] SAID, A., PEARLMAN, W.A. : A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, 1996.

[7] MALÝ, J. : Metoda pro kompresi obrazových signálů pomocí waveletové transformace, Diplomová práce, Ústav Telekomunikací FEKT VUT Brno, 2006. K dispozici na: http://wavelets.triablo.net/down/diplomova-prace.pdf

[8] ŘÍČNÝ, V., KRATOCHVÍL, T. : Základy televizní techniky – přednášky, Ústav radioelektroniky, Brno 2004

[9] SCHIMMEL, J. : Standardy pro přenos videa – přednášky předmětu Grafické a multimediální procesory, Ústav telekomunikací, FEKT VUT v Brně, 2007

[10] HANUS, S. : Bezdrátové a mobilní komunikace - skripta, Ústav Radioelektroniky, Brno 2005

[11] SZABÓ, Z. : Internet a zdravotnická informatika- přednáška, Katedra biomedicínské informatiky FBMI, ČVUT v Praze 2007, dostupné z URL http://www.izi.webz.cz/2pr_2007_08.pdf

[12] Video na PC, Portál o zpracování videa a audia : Seznam videokodeků dle FOURCC, dostupné z URL http://www.video.az4u.info

[13] PRCHAL, J., ŠIMÁK, B. : Digitální zpracování signálů v telekomunikacích, Vydavatelství ČVUT, Praha 2001

[14] RAJMIC, P. : Využití waveletové transformace a matematické statistiky pro separaci signálu a šumu - Dizertační práce, FEKT VUT v Brně, 2004.

[15] ZDENĚK GERLICKÝ, Z. : Srovnání používaných videokodeků na platformě PC - Bakalářská práce, České vysoké učení technické v Praze Fakulta elektrotechnická, Praha 2006

[16] ZÁVODNÝ, T. : Videoformáty, videokodeky – Bakalářská práce, Masarykova univerzita, Fakulta Informatiky, Brno 2003

[17] BODEČEK, K., DAŇĚČEK, P., VRBA, K. : Bezpečný JPEG2000, Fakulta elektroniky a komunikačních technologií VUT v Brně, Ústav telekomunikací, Datum publikování: 21.11.2006, dostupné z URL http://www.elektrorevue.cz/clanky/06047/index.html

[18] ZAPLATÍLEK, K., DOŇAR, B.: MATLAB - tvorba uživatelských aplikací, Praha: BEN - Technická literatura, ISBN 80-7300-133-0

[19] MISITI, M., MISITI, Y., OPPENHEIM, G., POGGI, J.,M. : Wavelet Toolbox for use with MATLAB®, MathWorks, 2002.

[20] Morgan Multimedia : http://www.morgan-multimedia.com

[21] Ligos Corporation : http://www.ligos.com

[22] MATLAB Central : 2-D lifting wavelet transform, dostupné z URL http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=11895&objectType=file

SEZNAM POUŽITÝCH ZKRATEK

ATSC	(Advanced Television Standards Committee)
AVC	(Advanced Video Coding) Standard pro kompresi digitálního videa H.264
B	1 Bajt (8 bitů)
CDF	(Cohen Daubechies Feauveau Wavelet) Biortogonální vlnka
CIF	(Common Intermediate Format) Standardní videoformát o velikosti obrazu 352 × 288 bodů
CWT	(Continuous Wavelet Transform) Spojitá vlnková trasformace
DCT	Diskrétní kosinová transformace
2D-DWT	(DWT2) Dvourozměrná DWT
DTWT	Diskrétní vlnková transformace s diskrétním časem
DWT	(Discrete Wavelet Transformation) Diskrétní vlnková transformace
EBCOT	(Embedded Block Coding with Optimized Truncation) Kompresní algoritmus použitý v JPEG2000
EZW	(Embedded Zerotree Wavelet) Metoda prahování koeficientů
FIR	Konečná impulsní odezva
FourCC	(Four Character Code) Identifikace kodeku
FT	Fourierova transformace
GOF	(Group Of Frame) Skupina snímků
HDTV	(High Definition TV) Obrazový formát
HDV	(High Definition Video) Obrazový formát
IEC	(International Electro-Technical Commission) Mezinárodní elektro – technická komise
IIR	Nekonečná impulsní odezva
ISO	(International Organization for Standardization) Mezinárodní organizace pro normalizaci
ITU-T	(International Telecommunications Union-Telecommunications standardization sector) Mezinárodní telekomunikační unie - Telekomunikační normalizační sektor
JPEG	(Joint Photographic Experts Group) Obrazový formát se ztrátovou kompresí
JPEG2000	(Joint Photographic Experts Group 2000) Standard pro počítačovou grafiku
LZW	(Lempel-Ziv-Welch) Bezeztrátová kompresní metoda
MPEG	(Motion Picture Experts Group) Metoda pro kódování videa

MSB	(Most Significant Bit) Nejvíce významný bit
MSE	(Mean Square Error) Střední kvadratická chyba
PSNR	(Peak Signal to Noise Ratio) Špičkový poměr signál - šum
Pixel	(Picture Element) Obrazový prvek
px	Pixel, jednotka obrazové informace
SNR	(Signal to Noise Ratio) Poměr signál - šum
SPIHT	(Set Partitioning In Hierarchical Trees) Metoda prahování koeficientů
STFT	Short-Time Fourierova transformace
SVC	(Scalable Video Coding)
QCIF	(Quarter Common Intermediate Format) Čtvrtinový CIF, standardní videoformát o velikosti obrazu 176 x 144 bodů
VLC	(Variable Length Coding) Kód s proměnnou délkou
YUV	(Yellow Under Violet) Způsob kódování obrazu, jeden pixel je vždy popsán třemi složkami - jasovou a dvěmi barevnými

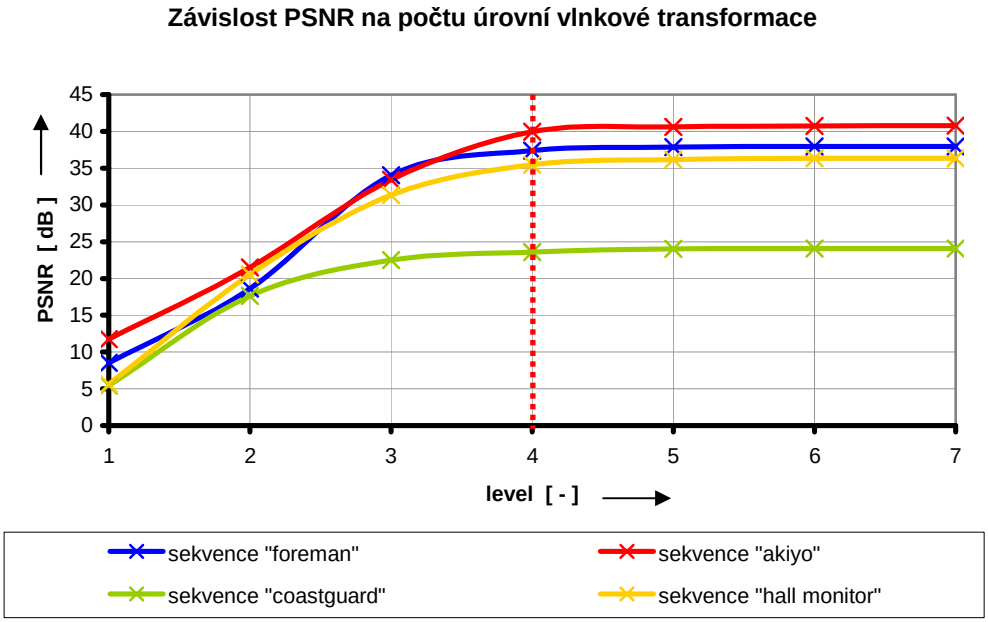
SEZNAM PŘÍLOH

PŘÍLOHY

A Ukázky prvních snímků z GOF testovacích sekvencí

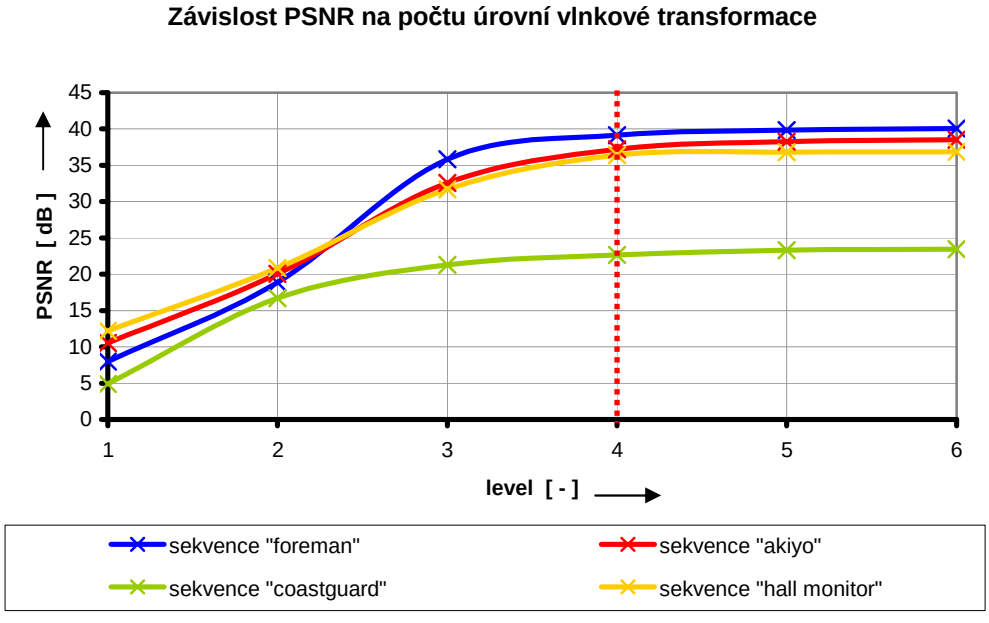
B Graf závislosti PSNR na počtu úrovní vlnkové transformace

B.1 Pro blok 128 pixelů



Poznámka: Komprese 0,5bit/px, vlnka = 'cdf97', velikost bloku 128 x 128,
kodér: 3DSPIHT s předzpracováním

B.2 Pro blok 64 pixelů



Poznámka: Komprese 0,5bit/px, vlnka = 'cdf97', velikost bloku 64 x 64,
kodér: 3DSPIHT s předzpracováním

C TABULKA NAMĚŘENÝCH HODNOT PŘI TESTOVÁNÍ NAVRŽENÉHO ALGORITMU

C.1 Tabulka naměřených hodnot při testování kodérů

		Scéna [-]					
	Kodér	**1**	**2**	**3**	**4**	**5**	**6**
PSNR [dB]	3D SPIHT	30,98	25,22	30,08	24,44	25,32	25,08
	3D SPIHT s předzpracováním	30,33	25,45	35,78	23,92	31,32	27,78
	2D SPIHT	30,56	24,87	29,05	24,05	24,68	24,49
	3D SPIHT s predikcí	29,98	24,63	35,41	23,95	31,39	27,60

Pozn. : Komprese 0,1bit/px, level = 6, vlnka = 'cdf97', velikost bloku 256 x 256

C.2 Tabulka naměřených hodnot při testování úrovně dekompozice

			Level [-]							
			8	**7**	**6**	**5**	**4**	**3**	**2**	**1**
PSNR [dB]	256 px	„foreman"	36,58	36,58	36,57	36,49	35,97	33,26	20,00	10,58
		„akiyo“	45,69	45,68	45,65	45,54	44,91	37,74	23,05	11,69
		„coastguard“	28,81	28,80	28,77	28,65	27,97	25,85	19,75	5,34
		„hall monitor“	38,53	38,52	38,47	38,29	37,25	32,03	15,59	4,83
	128 px	„foreman"		37,97	37,95	37,88	37,41	34,06	18,63	8,51
		„akiyo“		40,78	40,76	40,62	39,98	33,42	21,51	11,75
		„coastguard“		24,10	24,10	24,05	23,61	22,50	17,65	5,44
		„hall monitor“		36,36	36,35	36,19	35,47	31,34	20,56	5,61
	64 px	„foreman"			40,08	39,84	39,12	35,83	18,87	7,96
		„akiyo“			38,52	38,23	37,20	32,57	20,03	10,53
		„coastguard“			23,44	23,34	22,64	21,28	16,69	4,89
		„hall monitor“			36,86	36,82	36,41	31,71	20,87	12,20

Pozn. : Komprese 0,5bit/pixel, vlnka = „cdf97“, kodér 3D SPIHT s předzpracováním

C.3 Tabulka naměřených hodnot při testování druhu vlnky

	Testovací sekvence			
Vlnka [-]	**„foreman“**	**„coastguard“**	**„hall monitor“**	**„akiyo“**
	PSNR [dB]			
haar	25,22	24,79	37,30	35,36
db2	25,46	24,84	37,39	35,75
db3	25,55	24,90	37,49	35,66
db8	25,57	24,90	37,37	35,09
bior1.3	25,10	24,55	37,16	34,81
bior2.8	25,59	24,88	37,15	35,21
bior3.5	25,50	24,66	36,62	34,30
sym2	25,46	24,84	37,39	35,75
sym5	25,56	24,94	37,29	35,83
sym8	25,56	24,89	37,32	35,74
cdf97	42,79	28,78	40,36	43,84

Pozn. : Komprese 1bit/px, level = 6, velikost bloku 64 x 64, kodér: 3DSPIHT s předzpracováním

C.4 Tabulka naměřených hodnot při testování velikosti komprese

		Testovací sekvence			
	Komprese [bit/pixel]	**„foreman“**	**„coastguard“**	**„hall monitor“**	**„akiyo“**
		PSNR [dB]			
256 px	0,01	23,31	24,34	20,89	20,55
	0,02	25,17	27,19	23,43	21,77
	0,04	27,54	30,39	26,13	22,86
	0,06	28,83	32,94	28,03	23,47
	0,08	29,88	34,32	29,67	23,78
	0,10	30,33	35,78	31,32	23,92
	0,20	33,06	40,99	35,01	25,82
	0,40	35,44	44,87	37,31	27,64
	0,60	37,11	46,21	39,30	29,50
	0,80	38,76	47,81	40,17	29,95
	1,00	40,26	49,04	40,62	31,68
32 px	1,00	41,66	42,98	40,89	27,68
	2,00	45,09	47,66	44,38	32,64
	3,00	48,95	51,84	49	37,82
	4,00	53,86	57,87	53,76	42,94
	4,60	58,6	∞	58,17	46,11
	5,00	∞	∞	64,52	48,65
	5,50	∞	∞	∞	50,92
	6,00	∞	∞	∞	54,46
	7,00	∞	∞	∞	65,6
	8,00	∞	∞	∞	∞

Pozn. : Pro velikost bloku 256 x 256 level = 6, pro velikost bloku 32 x 32 level = 4, vlnka = 'cdf97', kodér: 3DSPIHT s předzpracováním

C.5 Tabulky naměřených hodnot při testování mezirámcové účinnosti kódování

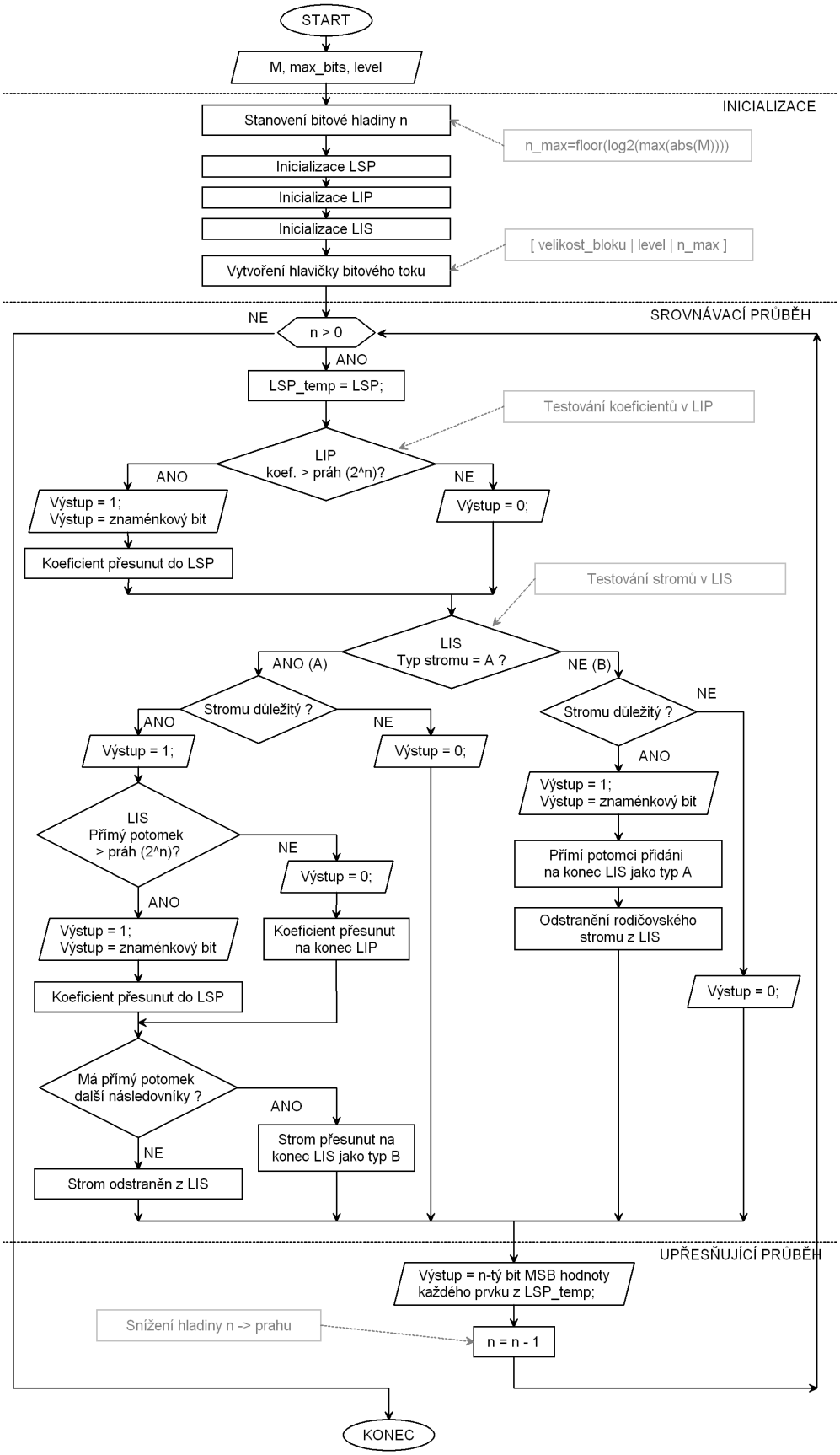
GOF [-]	Kodér	
	3D SPIHT s přdzpracováním	2D SPIHT
	PSNR [dB]	
1	35,01	27,50
2	34,79	27,45
3	32,40	27,16
4	31,54	27,12
5	31,45	27,15
6	31,02	27,25
7	31,61	27,45
8	31,88	27,40
9	32,32	27,33
10	32,12	27,18

Pozn. : Komprese 0,2bit/px, level = 6, vlnka = 'cdf97', velikost bloku 256 x 256, scéna: 'hall_monitor'

GOF [-]	Testovací sekvence			
	„foreman“	„coastguard“	„hall monitor“	„akiyo“
	PSNR [dB]			
1	33,06	26,30	35,01	40,99
2	30,21	26,31	34,79	38,59
3	29,82	26,08	32,40	35,92
4	29,75	25,96	31,54	36,20
5	29,54	25,74	31,45	35,97
6	30,78	25,39	31,02	37,84
7	30,18	25,10	31,61	39,26
8	29,44	24,43	31,88	37,64
9	30,77	23,78	32,32	36,25
10	30,60	23,57	32,12	36,49

Pozn. : Komprese 0,2bit/px, level = 6, vlnka = 'cdf97', velikost bloku 256 x 256, kodér: 3DSPIHT s předzpracováním

D VÝVOJOVÝ DIAGRAM ALGORITMU 3D SPIHT



E UKÁZKA ZDROJOVÉHO KÓDU FUNKCE K PROHLEDÁVÁNÍ STROMU

```
function value = func_FindDescMax(i, j, k, type, M,level)
%
% Najde nejvetsi hodnotu v absolutni hodnote potomku pixelu (i,j,k)
%
% input:    i ... radkova souradnice
%           j ... sloupcova souradnice
%           k ... snimkova souradnice
%           type ... typ stromu {0=A,1=B}
%           M ... vidosekvence, matice o rozmerech (i,j,k)
%
% output:   value ... nejvetsi absolutni hodnota
%
% autor:    Vojtech KINTL

value=0;
bandsize = 2.^(log2(size(M, 1)) - level + 1);
[s1,s2,s3] = size(M);
S = [];
index = 0; a = 0; b = 0;

while ((2*i-1)<s1 & (2*j-1)<s2 & (2*k-1)<s3)
    a = i-1; b = j-1;
    mind = [2*(a+1)-1:2*(a+2^index)];
    nind = [2*(b+1)-1:2*(b+2^index)];

    if (k==1)  %Pro stromy prvniho snimku
        zind = [1:(2^(index+1))];

    elseif (k==2) %Pro stromy druheho snimku
        if ((i<=bandsize/2)&(j<=bandsize/2))
            disp('Koren druheho snimku (cA koeficienty) ')
        else
            zind = [2^(index+1)+1:2*k*(2^index)];
        end

    elseif (k==3)   %Pro stromy tretiho snimku
        if ((i==1)&(j==1))
            index = 0; a = 0; b = 0;
            mind = [2*(a+1)-1:2*(a+2^index)];%[1 2]
            nind = [2*(b+1)-1:2*(b+2^index)];%[1 2]
            zind = [2^(index+1)+3+k*index:2^(index+1)+4+k*index];%[5 6]
            S = [S reshape(M(mind,nind,zind),1,[])];
            if type == 1
                S(:,1:8) = [];
            end
            value = max(abs(S));
            return
        else
            zind = [2^(index+1)+3+k*index:2^(index+1)+4+k*index];
        end

    elseif (k==4) %Pro stromy ctvrteho snimku
        if ((i==1)&(j==1))
            [s1,s2,s3] = size(M);
            S = [];
            index = 0; a = 0; b = 0;
            mind = [2*(a+1)-1:2*(a+2^index)];%[1 2]
            nind = [2*(b+1)-1:2*(b+2^index)];%[1 2]
            zind = [2^(index+2)+3:2^(index+2)+4];
            S = [S reshape(M(mind,nind,zind),1,[])];
            if type == 1
                S(:,1:8) = [];
            end
            value = max(abs(S));
            return
        else
            zind = [2^(index+2)+3:2^(index+2)+4];
        end
    end
```

```
    chk = mind <= s1;
    len = sum(chk);
    if len < length(mind)
        mind(len+1:length(mind)) = [];
    end

    chk = nind <= s2;
    len = sum(chk);
    if len < length(nind)
        nind(len+1:length(nind)) = [];
    end

    chk = zind <= s3;
    len = sum(chk);
    if len < length(zind)
        zind(len+1:length(zind)) = [];
    end

    S = [S reshape(M(mind,nind,zind),1,[])];
    index = index + 1;
    i = 2*a+1; j = 2*b+1;
end

if type == 1
    S(:,1:8) = [];
end

value = max(abs(S));
```

F UKÁZKA ZDROJOVÉHO KÓDU FUNKCE REALIZUJÍCÍ 3D SPIHT KÓDOVÁNÍ

```
function out= en3DSPIHT(M, max_bits, level)
% Matlab implementace 3D SPIHT algoritmu
% Koder
% input:    M ... 3D (caso - prostorova) matice
%           max_bits ... pocet pozadovanych bitu
%           level ... uroven prostorove vlnkove dekompozice
% output:   out ... vystupni bitovy tok
% Autor:    Vojtech KINTL

block_size=size(M,1);%prostorova velikost snimku
max_bits=round(max_bits);

%-------------------------------------------------------------------------
%----------- INICIALIZACE ------------------------------------------------
%-------------------------------------------------------------------------
bitctr = 0;
out = 2*ones(1,max_bits - 14);%vytvoreni vystupniho vektoru o delce
n_max =floor(log2(max(max(max(abs(M))))));%stanoveni bitove hladiny
Bits_Header = 0;
Bits_LSP = 0;
Bits_LIP = 0;
Bits_LIS = 0;

%-----------   stream header  ----------------------------------------------
%velikost obrazu, n_max,level
out(1,[1 2 3]) = [size(M,1) level n_max]; bitctr = bitctr + 24;
index = 4;
Bits_Header = Bits_Header + 24;

% ----------- inicializace seznamu LIP, LIS, LSP --------------------------
% LIP – List of Insignificant Pixels, seznam insignifikantních koeficientu
% LIS – List of Insignificant Sets, seznam insignifikantních skupin
% LSP – List of Significant Pixels, seznam signifikantních koeficientu

%Inicializace LIP.....................................................
temp = [];
%velikost nejvyssi urovne dekompozice
bandsize = 2.^(log2(size(M, 1)) - level + 1);
temp1 = 1 : bandsize;
for i = 1 : bandsize
    temp = [temp; temp1];
end
LIP(:, 1) = temp(:);
temp = temp';
LIP(:, 2) = temp(:);
tLIP=LIP;
LIP=[LIP;LIP];
LIP=[LIP;LIP];
LIP(1:bandsize^2, 3) = 1;
LIP(bandsize^2+1:(bandsize^2)*2, 3) = 2;
LIP((bandsize^2)*2+1:(bandsize^2)*3, 3) = 3;
LIP((bandsize^2)*3+1:end, 3) = 4;

%Inicializace LIS.....................................................
tLIP(1:bandsize^2, 3) = 1;
LIS(:, 1) = tLIP(:, 1);
LIS(:, 2) = tLIP(:, 2);
LIS(:, 3) = tLIP(:, 3);

%Odstraneni CA koficientu prvniho a druheho snimku ze seznamu
pstart = 1;
pend = bandsize / 2;
for i = 1 : bandsize / 2
    LIS(pstart : pend, :) = [];
    pdel = pend - pstart + 1;
    pstart = pstart + bandsize - pdel;
    pend = pend + bandsize - pdel;
end
tLIS=LIS;
tLIS(:, 3)=2;
LIS=[LIS ; tLIS];
LIS=[LIS ;LIP(2*(bandsize^2)+1:end, :) ];
LIS(:, 4) = zeros(length(LIS(:, 1)), 1);
clear tLIS tLIP;
```

```
%Inicializace LSP.....................................................
LSP = [];

n = n_max;

%-------------------------------------------------------------------------
%----------- ENCODING ----------------------------------------------------
%-------------------------------------------------------------------------
%while(n > 0)
while(bitctr < max_bits)

    % Sorting Pass ------------------------------------------------------
    %-------------------------------------------------------------------

    % LIP pass ----------------------------------------------------------
    LIPtemp = LIP; temp = 0;
    for i = 1:size(LIPtemp,1)
        temp = temp+1;
        if (bitctr + 1) >= max_bits
            if (bitctr < max_bits)
                out(length(out))=[];
            end
            return
        end

        % significance? 1: positive; 0: negative
        if abs(M(LIPtemp(i,1),LIPtemp(i,2),LIPtemp(i,3))) >= 2^n
            % output -> 1
            out(index) = 1; bitctr = bitctr + 1;
            index = index +1;

            %sign 1: positive; 0: negative
            sgn = M(LIPtemp(i,1),LIPtemp(i,2),LIPtemp(i,3))>=0;
            out(index) = sgn; bitctr = bitctr + 1;
            index = index +1;

            % into LSP
            LSP = [LSP; LIPtemp(i,:)];

            % LIP remove
            LIP(temp,:) = []; temp = temp - 1;
        else
            % output -> 0 (not significant)
            out(index) = 0; bitctr = bitctr + 1;
            index = index +1;
        end
    end

    % LIS pass ----------------------------------------------------------
    LIStemp = LIS; temp = 0; i = 1;
    while ( i <= size(LIStemp,1))
        temp = temp + 1;

        % LIS is entry type A - potomci daného prvku
        if LIStemp(i,4) == 0
            if bitctr >= max_bits
                return
            end

            % returns highest value from the subtree
               max_d = func_FindDescMax(LIStemp(i,1), LIStemp(i,2),...
               LIStemp(i,3),LIStemp(i,4), M,level);
            % is there significance?
            if max_d >= 2^n
                % yes - output -> 1
                out(index) = 1; bitctr = bitctr + 1;
                index = index +1;
                x = LIStemp(i,1); y = LIStemp(i,2); z = LIStemp(i,3);

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
                end
```

```
% 1 top-left front significance---------------------------
if abs(M(2*x-1,2*y-1,2*z-1)) >= 2^n
    % YES - into LSP, output -> 1 and sign
    LSP = [LSP; 2*x-1 2*y-1 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x-1,2*y-1,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    % NOT - into LIP, output -> 0
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x-1 2*y-1 2*z-1];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end

% 2 top-right front significance--------------------------
if abs(M(2*x-1,2*y,2*z-1)) >= 2^n
    LSP = [LSP; 2*x-1 2*y 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x-1,2*y,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x-1 2*y 2*z-1];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end

% 3 bottom-left front significance------------------------
if abs(M(2*x,2*y-1,2*z-1)) >= 2^n
    LSP = [LSP; 2*x 2*y-1 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x,2*y-1,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x 2*y-1 2*z-1];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end

% 4 bottom-right front significance-----------------------
if abs(M(2*x,2*y,2*z-1)) >= 2^n
    LSP = [LSP; 2*x 2*y 2*z-1];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x,2*y,2*z-1)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x 2*y 2*z-1];
```

```
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end


% 5 top-left back significance----------------------------
if abs(M(2*x-1,2*y-1,2*z)) >= 2^n
    % YES - into LSP, output -> 1 and sign
    LSP = [LSP; 2*x-1 2*y-1 2*z];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x-1,2*y-1,2*z)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    % NOT - into LIP, output -> 0
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x-1 2*y-1 2*z];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end

% 6 top-right back significance---------------------------
if abs(M(2*x-1,2*y,2*z)) >= 2^n
    LSP = [LSP; 2*x-1 2*y 2*z];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x-1,2*y,2*z)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x-1 2*y 2*z];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end

% 7 bottom-left back significance-------------------------
if abs(M(2*x,2*y-1,2*z)) >= 2^n
    LSP = [LSP; 2*x 2*y-1 2*z];
    out(index) = 1; bitctr = bitctr + 1;
    index = index +1;
    sgn = M(2*x,2*y-1,2*z)>=0;
    out(index) = sgn; bitctr = bitctr + 1;
    index = index +1;
else
    out(index) = 0; bitctr = bitctr + 1;
    index = index +1;
    LIP = [LIP; 2*x 2*y-1 2*z];
end

if (bitctr + 1) >= max_bits
    if (bitctr < max_bits)
        out(length(out))=[];
    end
    return
end
```

```
            % 8 bottom-right back significance------------------------
            if abs(M(2*x,2*y,2*z)) >= 2^n
                LSP = [LSP; 2*x 2*y 2*z];
                out(index) = 1; bitctr = bitctr + 1;
                index = index +1;
                sgn = M(2*x,2*y,2*z)>=0;
                out(index) = sgn; bitctr = bitctr + 1;
                index = index +1;
            else
                out(index) = 0; bitctr = bitctr + 1;
                index = index +1;
                LIP = [LIP; 2*x 2*y 2*z];
            end



            % space for more descendants?
            if ((2*(2*x)-1)<size(M,1)&(2*(2*y)-1)<size(M,2)&((2*2*z)<=size(M,3)))
                % YES - [x,y] into LIS, LIStemp as type B entry
                LIS = [LIS; LIStemp(i,1) LIStemp(i,2) LIStemp(i,3) 1];%typ B
                LIStemp = [LIStemp; LIStemp(i,1) LIStemp(i,2) LIStemp(i,3) 1];
            end
            % remove [x,y,z typ] from LIS
            LIS(temp,:) = []; temp = temp-1;

        else %neni zadny potomek ve stromu dulezity
            % no significance, output -> 0 and no changes
            out(index) = 0; bitctr = bitctr + 1;
            index = index +1;
        end


    else
  % type B entry in LIS - vsichni potomci prvku krome 8 bezprostredne nasledujicich
        if bitctr >= max_bits
            return
        end

        % maximal value from grandchildren
           max_d = func_FindDescMax(LIStemp(i,1), LIStemp(i,2),...
           LIStemp(i,3),LIStemp(i,4), M,level);

        % do they have significance?
        if max_d >= 2^n
            % YES - output -> 1, add their parents into LIS, LIStemp as A type
            out(index) = 1; bitctr = bitctr + 1;
            index = index +1;
            x = LIStemp(i,1); y = LIStemp(i,2); z = LIStemp(i,3);
            LIS = [LIS;     2*x-1 2*y-1 2*z-1 0;2*x-1 2*y 2*z-1 0; ...
                2*x 2*y-1 2*z-1 0; 2*x 2*y 2*z-1 0
                2*x-1 2*y-1 2*z 0;2*x-1 2*y 2*z 0; ...
                2*x 2*y-1 2*z 0; 2*x 2*y 2*z 0];
            LIStemp = [LIStemp; 2*x-1 2*y-1 2*z-1 0;2*x-1 2*y 2*z-1 0; ...
                2*x 2*y-1 2*z-1 0; 2*x 2*y 2*z-1 0
                2*x-1 2*y-1 2*z 0;2*x-1 2*y 2*z 0; ...
                2*x 2*y-1 2*z 0; 2*x 2*y 2*z 0];

            % remove [x,y] from LIS
            LIS(temp,:) = []; temp = temp - 1;
        else
            % NO - output -> 0
            out(index) = 0; bitctr = bitctr + 1;
            index = index +1;
        end
    end

    i = i+1;

end
```

```
    % Refinement Pass ----------------------------------------------------
    %--------------------------------------------------------------------

    temp = 1;
    value = floor(abs(2^(n_max-n+1)*M(LSP(temp,1),LSP(temp,2),LSP(temp,3))));

    % as long as there are items in LSP compliant with the following condition
    while (value >= 2^(n_max+2) & (temp <= size(LSP,1)))
        if bitctr >= max_bits
            return
        end
        %C = bitget(A, bit) returns the value of the bit at position bit in A
        s = bitget(value,n_max+2);
        % save MSB
        out(index) = s; bitctr = bitctr + 1;
        index = index +1;
        temp = temp + 1;
        if temp <= size(LSP,1)
            value = floor(abs(2^(n_max-n+1)*M(LSP(temp,1),LSP(temp,2),LSP(temp,3))));
        end
    end

    n = n - 1;

end
```

G OBSAH PŘILOŽENÉHO CD

- Adresá *matlab*, kde jsou všechny zdrojové kódy (m-file) a testovací sekvence ve formátu yuv,
- adresá *text* s textem této diplomové práce ve formátu pdf,
- adresář *metadata*, popisný soubor diplomové práce ve formátu pdf,
- adresář *zaloha*, který obsahuje výše uvedené adresáře zkomprimované do archivu zip.